International Science and Technology Journal
المجلة الدولية للعلوم والتقنية
Volume 31 العدد
November 2022 نوفمبر
المجلة الدولية للعلوم والتقنية
ISTJ

Ethereum Blockchain-based System Design and Prototyping of Libyan Dinar Currency

1.Ali Farij Kaeib ,2. Abdulrahman B. Alhaaj Mohammed, 3.Suhayb B. Alhaaj Mohammed, 4.Osama Aboubaker Salem Abourodes,5. Ramadan Amer Ali Emhamed

*Corresponding author: E-mail addresses: ALI.KAEIB@sabu.edu.ly, (Second author: aalbeshti@gmail.com , (Third author) suhibelbeshti@gmail.com ,(fourth author) abourods.oas@gmail.com , (Fifth author) ramadan.amer1133@gmail.com

Abstract: The Libyan Dinar token (LDT) is introduced in this paper as, a stable digital currency subject to the (ERC-20) standard on the Ethereum blockchain, to be issued by a trusted financial institution and supported by the Libyan dinar as a reference currency so that the value of the digital currency remains. It is always equal to the Libyan dinar. the concept of a stable currency and what is the Ethereum network were discussed. The following steps of creating and dealing with the contract that creates the currency. Finally, discuss the advantages that digital currency gives to users and merchants when using it as an alternative to fiat currencies. some potential obstacles are mentioned in this paper and suggested solutions.

Keywords: Token; Stablecoin; Etherume; Smart contract; Solidity; Electronic payments; Fintech,

الملخص

في هذه الورقة نقترح عملة الدينار الليبي الرقمي (الرمزي) (دل ر), عملة رقمية مستقرة تخضع لمعيار **(ERC−20)** علي سلسلة الكتل ايثيريوم, ليتم إصدارها بواسطه مؤسسة مالية موثوقة و دعمها بالدينار الليبي كعملة مرجعيه بحيث تضل قيمة العملة الرقمية دائما مساويه للدينار الليبي, نناقش في الورقة مفهوم العملة المستقرة و ماهية شبكة الايثيريوم, ونتابع لخطوات انشاء والتعامل مع العقد المنشئ للعملة, أخيرا نناقش المزايا التي تمنحها العملة الرقمية للمستخدمين والتجار عند استعمالها كبديل للعملات الورقية,

وناقش أيضا بعض العقبات المحتملة للمشروع و حلول مقترحة.

الكلمات المفتاحية: عملة رمزية، عملة مستقرة، عقد ذكي، سوليديتي، دفع رقمي، تقنية مالية.

INTRODUCTION

The idea for asset-pegged cryptocurrencies was initially popularized in the Bitcoin community by the white paper authored by J.R. Willett in Jan 2012. Today, an increasing number of stablecoins have already been developed and deployed on multiple blockchains either on second layers like omni on bitcoin or on smart contract supporting chains such as Ethereum, Solana, Tron, Binance smart chain, and many others.

Using blockchain technology, specifically the smart contract specification of the Ethereum blockchain, demonstrate how can a digital (Libyan Dinar-backed) currency be used as an alternative to physical cash for almost every aspect of daily life financial transacting.

Ethereum Virtual Machine

The EVM is a straightforward stack-based architecture. The machine's word size (and thus the size of stack items) is 256 bits. This was selected to make the Keccak256 hash scheme and elliptic-curve computations easier [1]. The memory model is a straightforward word-addressed byte array. The stack can hold up to 1024 items. The machine also has an independent storage model, which is similar to memory in concept but is a word-addressable word array rather than a byte array. Storage, as opposed to memory, is nonvolatile and is kept as part of the system state. All storage and memory locations are initially defined as zero. The machine does not adhere to the traditional von Neumann architecture. Rather than storing programme code in widely accessible memory or storage, it is kept separate in a virtual ROM that can only be accessed via a specialised instruction.[2]

The machine can have exceptional execution for several reasons, including stack underflows and invalid instructions. They, like

the out-of-gas exception, do not preserve state changes. Instead, the machine immediately stops and reports the problem to the execution agent (either the transaction processor or, recursively, the spawning execution environment), which will handle it separately. [2]

World state (State)

In Ethereum, the world state (state) is a mapping between addresses (160-bit identifiers) and account states (a data structure serialised as RLP). Although this mapping is not stored on the blockchain, it is assumed that the implementation will keep it in a modified Merkle Patricia tree[3]. The tree necessitates the use of a simple database backend that keeps a mapping of byte arrays to byte arrays; this underlying database is known as the state database. This has several advantages. For starters, because the root node of this structure is cryptographically dependent on all internal data, its hash can be used as a secure identity for the entire system state. Second, because it is an immutable data structure, it can return to any previous state (whose root hash is known)
to be recalled by simply altering the root hash accordingly. Because all such root hashes are stored in the blockchain, it easily revert to previous states. Figure. 1 depicts the world state tree in Ethereum[4].

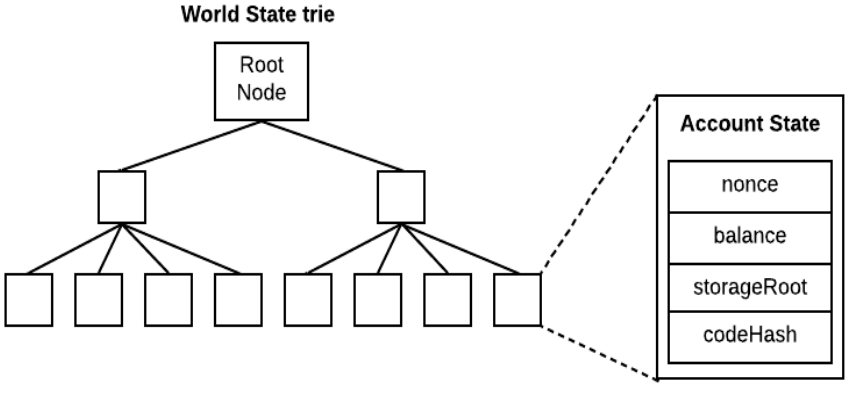

Figure .1 Ethereum state tree [4]

As shown, the account state, **σ**[a], comprises the following four fields:

nonce: A scalar value equal to the number of transactions sent from this address or, in the case of accounts with associated code, the number of contract-creations made by this account. For account of address a in state **σ**, this would be formally denoted $\boldsymbol{\sigma}[a]_n$.

balance: A scalar value equal to the number of Wei owned by this address. Formally denoted $\boldsymbol{\sigma}[a]_b$.

storageRoot: A 256-bit hash of the root node of a Merkle Patricia tree that encodes the storage contents of the account (a mapping between 256-bit integer values), encoded into the tree as a mapping from the Keccak 256-bit hash of the 256-bit integer keys to the RLP-encoded 256-bit integer values. The hash is formally denoted $\boldsymbol{\sigma}[a]_s$.

codeHash: The hash of this account's EVM code—this is the code that is executed if this address receives a message call; it is immutable and thus cannot be changed after construction, unlike all other fields. For later retrieval, all such code fragments are stored in the state database under their corresponding hashes. This hash is formally known as $\boldsymbol{\sigma}[a]_c$. [4]

Smart Contracts and Solidity

As previously stated, there are two kinds of accounts in Ethereum: externally owned accounts (EOAs) and contract accounts. Users control EOAs, often through software such as a wallet application that is not part of the Ethereum platform. Contract accounts, on the other hand, are managed by programme code (also known as "smart contracts") that is executed by the Ethereum Virtual Machine. In short, EOAs are simple accounts with no associated code or data storage, whereas contract accounts have both.[5]

MATERIALS AND METHODS

Contract source code is written in solidity, an object-oriented programming language for writing smart contracts. It is used for implementing smart contracts on various blockchain platforms, most notably, Ethereum. Editing and compiling were carried out through Remix IDE, an open-source web and desktop application for the development and testing of smart contracts.

The contract was deployed on the Goerli test network instead of the Ethereum main net, this is to eliminate the cost of transactions since it is only intended for demonstration. Transactions for contract creation and confirming token minting/burning procedures were sent from the metamask browser plugged in a wallet, (Remix supports injected web3 execution environment choice on deployment).

ERC-20 standard for smart contracts

The ERC-20 (Ethereum Request for Comments 20) Token Standard, proposed in November 2015 by Fabian Vogelsteller, is a Token Standard that implements an Application Programming Interface (API) for tokens within Smart Contracts.[6]

It allows you to transfer tokens from one account to another, get an account's current token balance, and see the total supply of the token available on the network. It can also grant permission for a certain amount of tokens from an account to be spent by a third-party account.

A smart contract must implement certain methods and events in order to be called an ERC-20 Token Contract. And, once deployed, it will be in charge of keeping track of the tokens created on Ethereum [7].

DISCUSSION

Ethereum: Several variants of blockchains exist. While Bitcoin still remains the most famous one by the public, Ethereum probably represents one of the most interesting solutions. This is due to the fact that Ethereum provides a vast range of use case applications enabled by smart contracts. Ethereum is often described with the term “world computer”, since this platform enables running distributed applications (i.e. smart contracts) in a distributed manner. It provides a way to create self-executing and self-enforcing contracts. Their execution is triggered via transactions. Once generated, nodes in the P2P system execute the related code. This causes a change of the state. All this is recorded in the blockchain. Thus, through the blockchain all nodes synchronize their replicated state globally, in a manner that is fully verifiable by

any system participant. That is why the distributed code run on the blockchain is referred as a smart contract. Once deployed, it cannot be modified. Hence, parties, that agree on the use of this code, are aware that there is no possibility to breach the agreement. (They can, of course, decide to not use that contract anymore, if for some reason the contract becomes obsolete.) [8]

In Ethereum, smart contracts are considered internal accounts, that can interact among themselves and with externally owned accounts, which are in fact users that employ the system. Both these kinds of accounts have their own balance, expressed in a distributed currency referred as Ether [9]. The Ether is the fuel for operating in Ethereum. Every transaction in Ethereum is made possible through a payment made by the clients of the platform to the machines executing the requested operations [10]. This enables several applications, ranging from the exchange of cryptocurrencies, to financial applications, storing and management of tokens and digital assets, notary systems, identity management, voting systems, up to those application that require the traceability of resources and assets. Several works find many application domains in healthcare, supply chain, Internet of Things, etc. [11]

Stablecoins: stablecoins are non-interest-bearing coins designed to have stable value against a reference currency or asset[12]. Stability is achieved through two commitments. First, the issuer agrees to mint and buy back coins at par. Second, the issuer holds assets to back its obligation to redeem the outstanding stablecoins. This "reserve" provides comfort that the issuer can buy back all outstanding coins, on demand. Reserve assets should be denominated in the currency of the reference asset, remain highly liquid during a crisis, and incur extremely small losses in a run or stressed market conditions. [13]

LDT stablecoin:

LDT project scheme has 3 stages:

1. Ethereum blockchain, all token transactions are recorded in the Ethereum blockchain through an ERC20 complaint smart contract.
2. Second stage being the contract itself, which, can create (mint) or destroy (burn) LDT tokens as instructed through owner, monitor

and record all token transactions, current supply etc.
3. Lastly, identify this contract owner as a trusted financial institution to be responsible for issuing and redeeming LDTs and accepting corresponding LYD deposits or making corresponding LYD withdrawals to/from cash reserves that back the value of LDTs in circulation. See figure .2

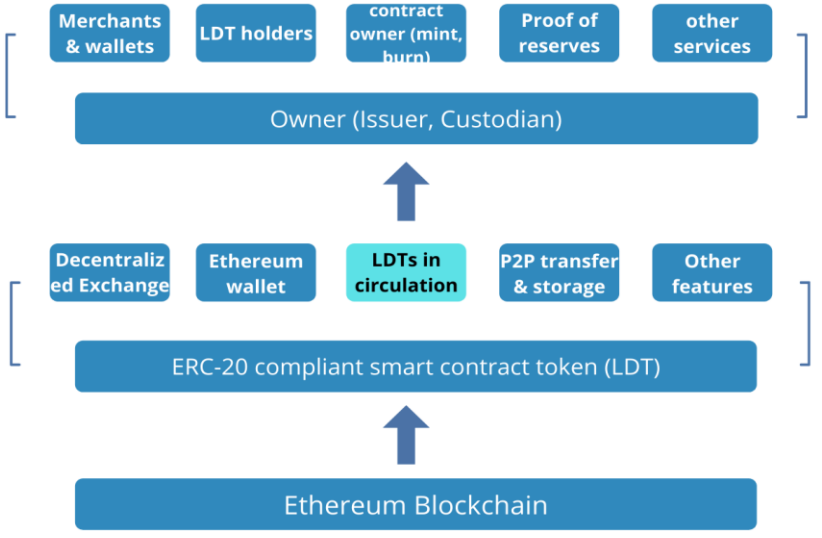

Figure 2. LDT scheme

Flow of Funds:
Figure. 3 demonstrates how LDTs circulate.
1. User deposits LYD currency into Business's bank account.
2. Business generates and credits the user's LDT account. LDTs enter circulation. Amount of LYD currency deposited by user = amount of LDTs issued to user.
3. Users transact with LDTs. The user can transfer, exchange, and store LDTs on Ethereum.
4. The user deposits LDTs into business's address for redemption into LYD.
Business destroys the LDTs and sends LYD currency to the ser's bank account, or in cash

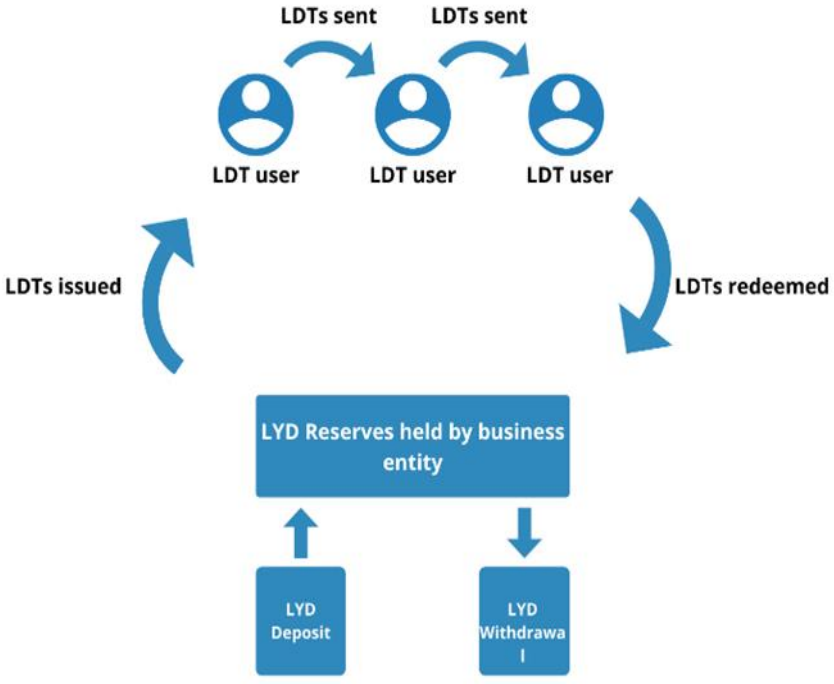

Figure. 3 Flow of funds

Token Contract:

The source code contains several parts:

1. Line 1 pragma solidity 0.5.16; defines compiler version.
2. Lines 2 to 27 define ERC-20 interface standard
3. Lines 28 to 40 define Context contract for our contract to inherit from.
4. Lines 41 to 91 contain SafeMath library functions.
5. Lines 92 to 127 contain Ownership Module (ownable) for our contract to inherit ownership properties.
6. Starting with line 128, main contract begins, calling inherited contracts, defining constructor function (setting token name, symbol, supply…), mintin and burning functions, and some other functions for transfers, approvals and allowances. (Please access code here: https://gist.github.com/AAlbeshti/00e8c4df6d50c4320fa619c083a98dae.

Deployment and functions testing

After having completed writing and editing our contract code on Remix, followed these steps were followed to deploy our contract:

1. Save our contract as LibyanDinarToken.sol.
2. Compile the contract using a compatible solidity compiler from compilers list.
3. Select our deployment environment as "injected Web3", this prompts remix to connect to our metamask account from Chrome extensions.
4. Remix then sets the deployment account to our metamask address, meaning that this will be the account sending the contract creation transaction, effectively becoming the contract owner account.
5. Clicking deploy prompts metamask to require approval for the transaction, and the contract is deployed into the Ethereum Blockchain as shown in figure 4.

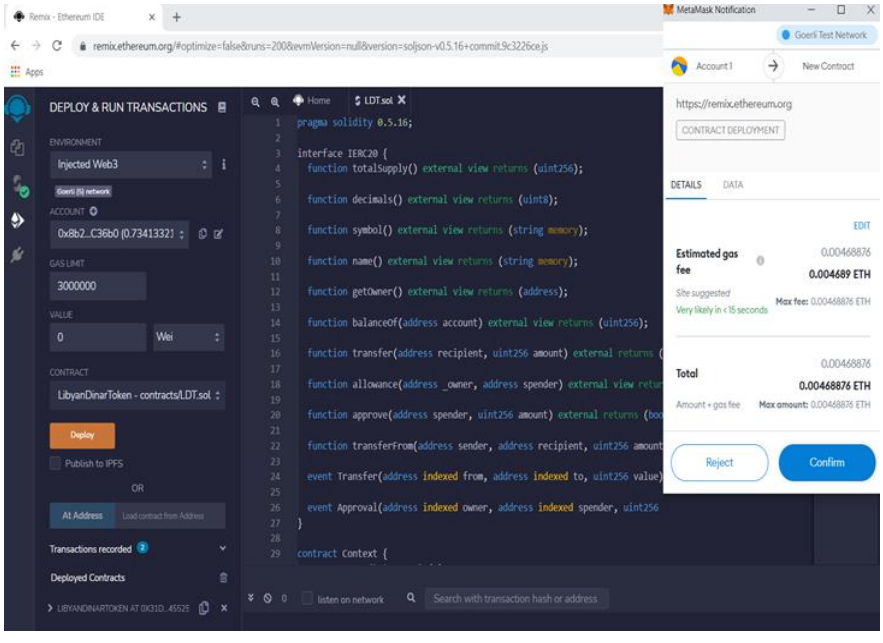

Figure .4 Contract deployment on Goerli test network

Contract creation transaction details could be also derived from goerli testnet explorer, Figure 4. show this info:

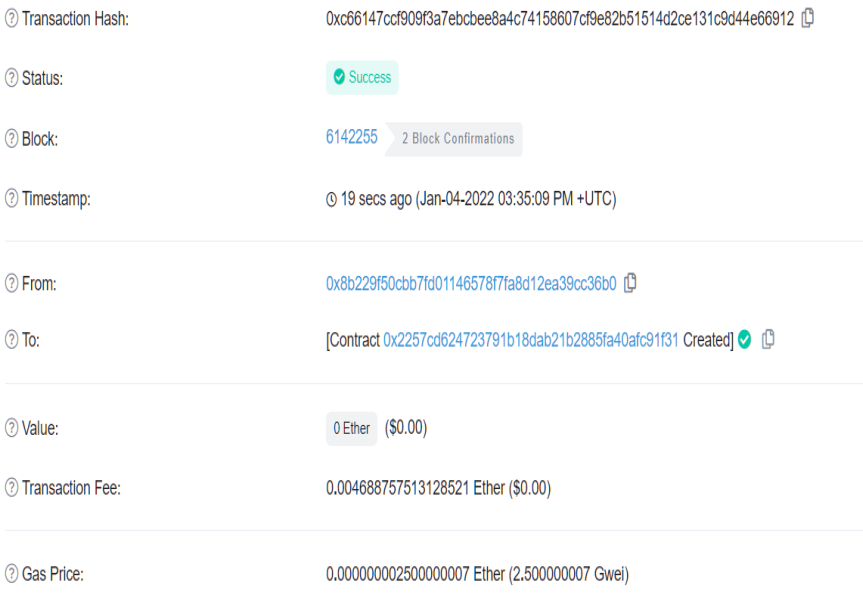

Figure 5. Contract creation transaction on etherscan.io explorer

As it can be seen, contract: 0x2257cd624723791b18dab21b2885fa40afc91f31 was created from address: 0x8b229f50cbb7fd01146578f7fa8d12ea39cc36b0, through transaction (Txhash: 0xc66147ccf909f3a7ebcbee8a4c74158607cf9e82b51514d2ce131c9d44e66912).
Initial token supply, number of accounts holding the token and total token transactions number could also be retrieved from the contract token page, as shown in Figur 6.

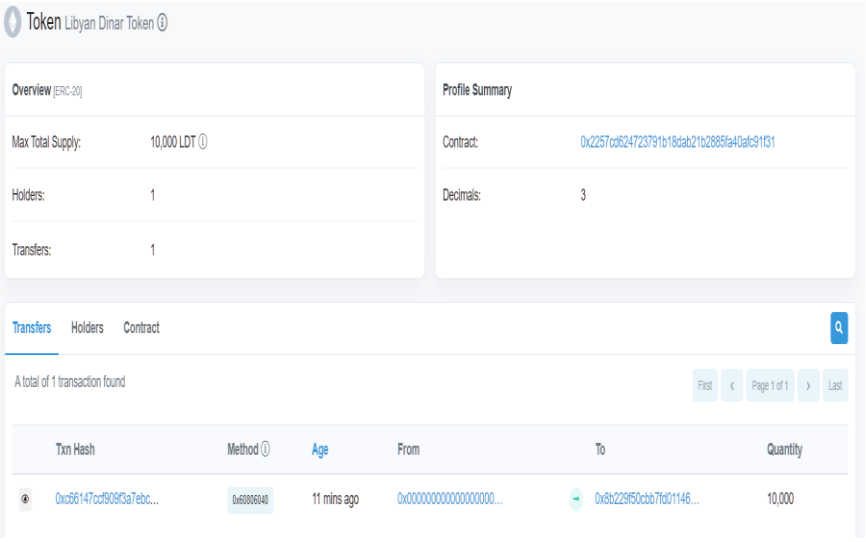

Figure 6-contract token info: total supply, holders, transfers

Minting and burning

Interacting with a contract requires:

1. Having its full code.
2. Satisfying caller's requirement for the specific function called within the contract.

In our contract, minting new tokens can only be performed from the owner's address. However, any account can burn tokens in its possession.

Minting new tokens initiate by sending a transaction containing the minting parameters to the contract address, contract then sends the tokens to the owner's address.

burning tokens can be performed by either calling the burn function from the contract or by sending the tokens to the ***zero address***. (Note that sending tokens to the ***zero address*** does not modify ***totalSupply***).

Interacting with the contract is possible within Remix IDE, simply compile the contract code, then enter the contract's address from the deployment section next to ***At Adress***, you will be introduced to

every function available within the contract that can be interacted with, Figure 7 illustrates a minting process, Figure 8, 9 show minting transaction details on etherscan, and the resulting change to total supply.

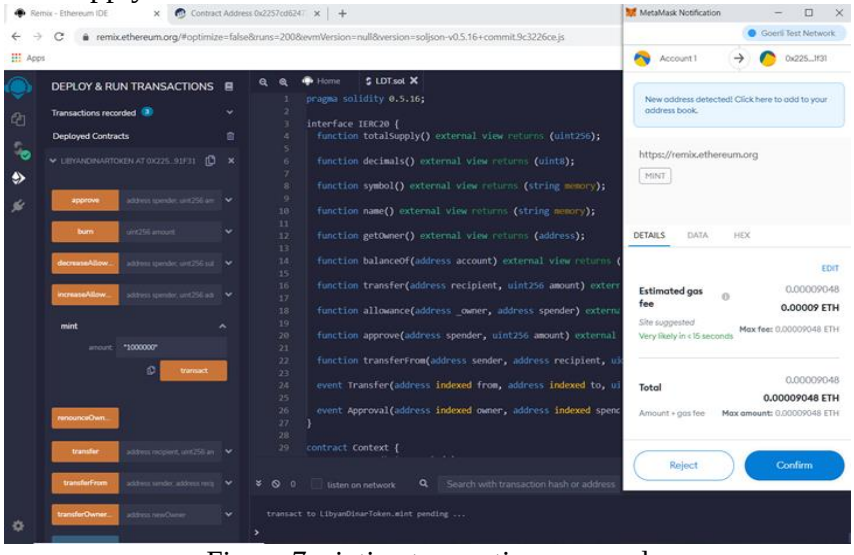

Figure *7* minting transaction approval

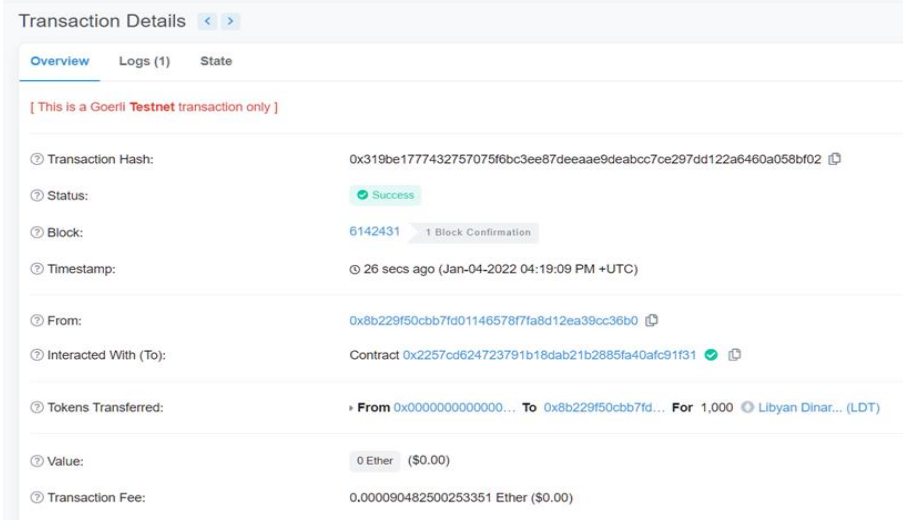

Figure .8 minting transaction details

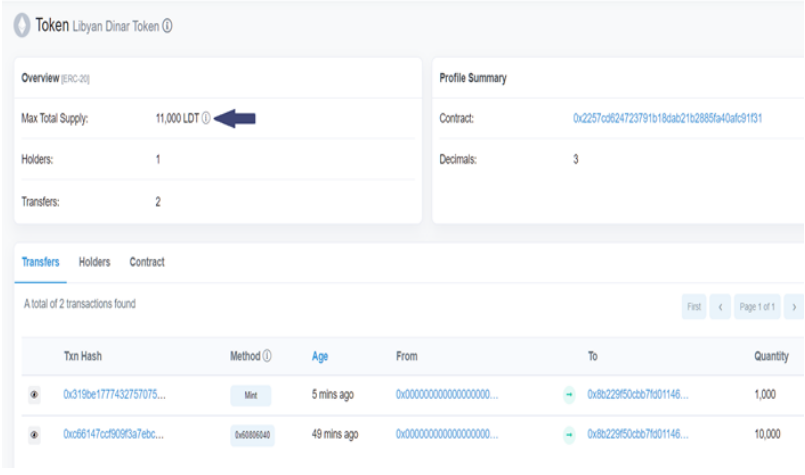

Figure .9 change to total supply on etherscan.io

Through goerli testnet explorer (goerli.etherscan.io), able to monitor state changes and transactions related to our token and perform minting burning or requesting any data required. first, let's verify our contract source code:
The contract has been navigated to an address, and right above the list of transactions, select contract, and then click "verify and publish" as shown in Figure 10.

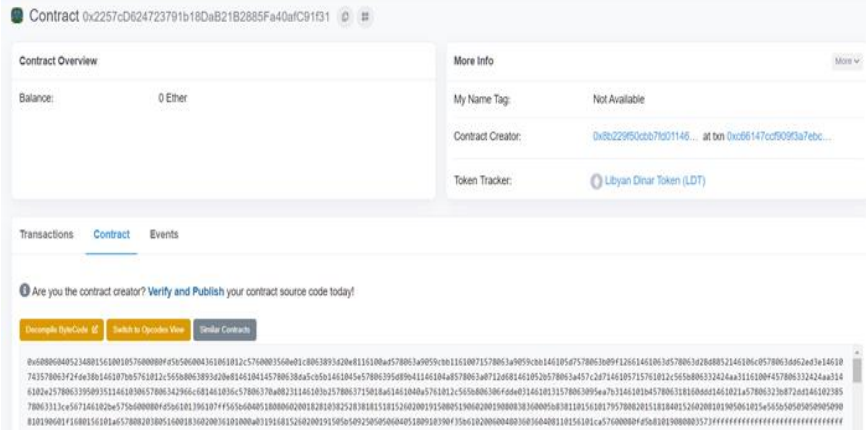

Figure .10 verifying the contract source code (step 1)

On this page, as illustrated in the figure. 11 the request has been confirmed and then select the compiler version and licence type.

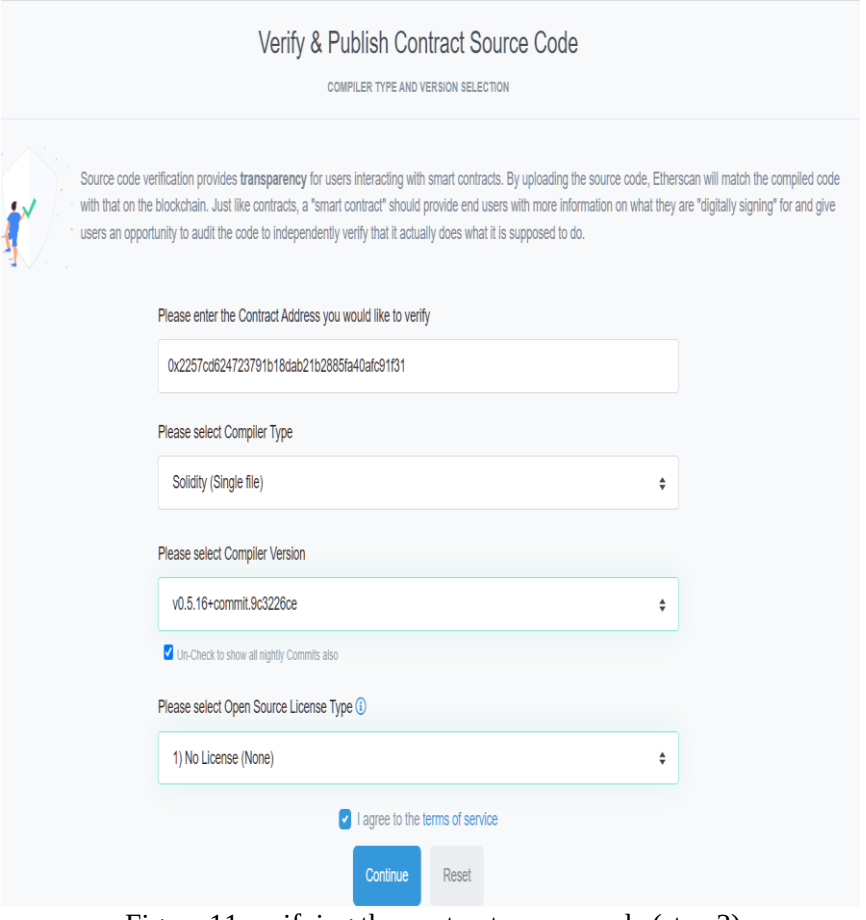

Figure 11 verifying the contract source code (step 2)

The page has been redirected to enter the contract code and finally select "verify and publish"as in Figure 12, 13.

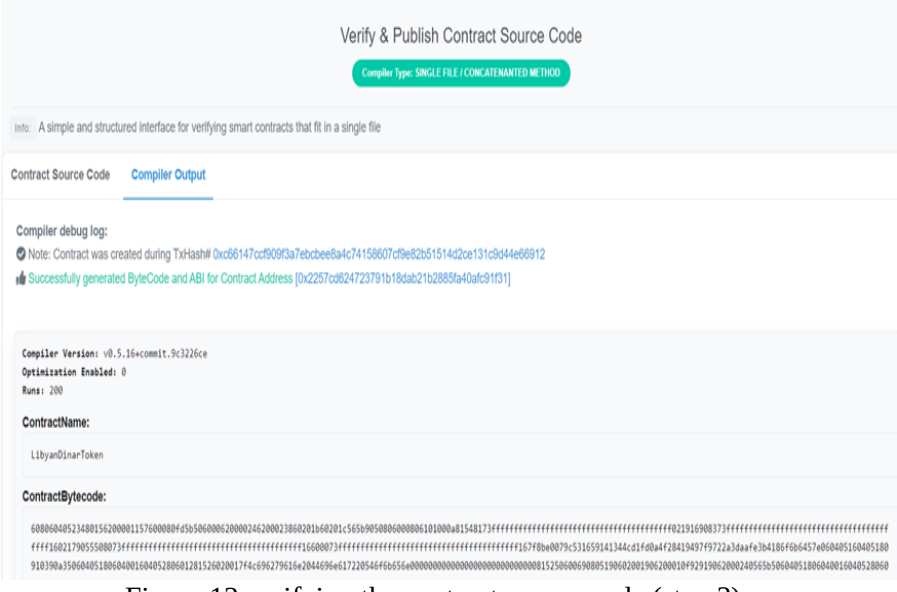

Figure 12 verifying the contract source code (step 3)

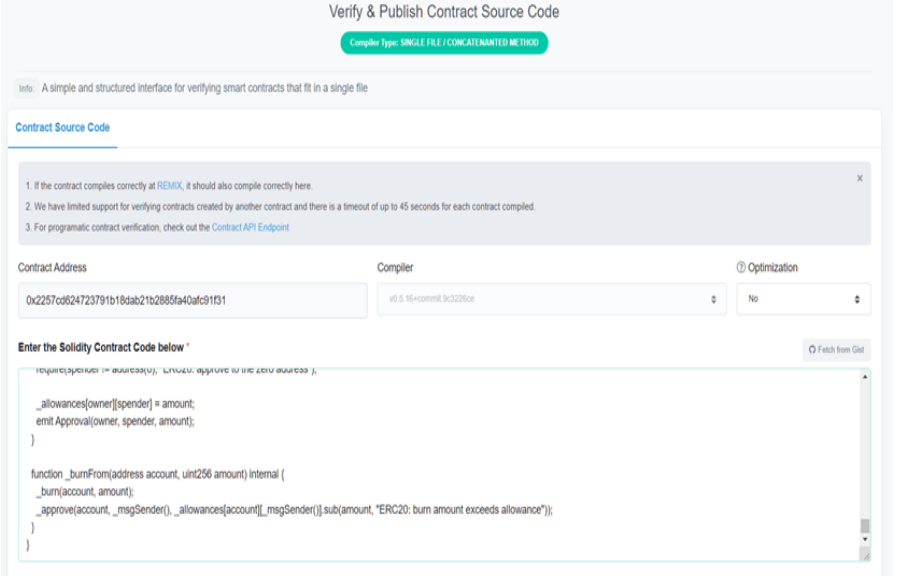

Figure 13 successful source code verification

Transacting and monitoring

After validating the source code, one can use etherscan explorer to retrieve any available information from the contract (read contract) or call contract functions (write contract), depending on the caller's permissions.

CONCLUSION

Libyan Dinar token is designed to leverage the new innovations of blockchain technology to improve the function of money, while being supported by traditional infrastructure that can ensure it is trustworthy.

Issued by a regulated trust company or financial institution, LDTs can offer a solution that combines the trust and stability of fiat currency with the utility and immediacy of digital assets, where the token substantiates to be easier to exchange than physical cash while maintaining the same unit of account and providing the same store of value.

Some of the features and benefits of using LDT include efficient operating and short redemption windows, unlimited storage capacity, cheap and fast transacting, 24/7 availability, limited physical contact, safety and anonymity.

And above all, when widely supported and accepted, LDT can help eliminate cash shortages immensely and feasibly by either being offered as an alternative for cash for salaries and payouts through banks -must be through means supported by CBL like a CBDC, since this can cause massive disruptions to monetary policy or loss of control over money supply-, or being issued by trusted financial institutions for cheque deposits and bank transfers instead of cash.

There are however, some impediments to the project that could limit its potential or even cause relative failure, these include:

1. Lacking economic incentive for local institutions to run the project, because generating any revenue through lending or interest-bearing products is prohibited.
2. recurrent Ethereum network congestions make transactions unfeasibly expensive.
3. Transacting LDT tokens requires Eth for gas payment, which complicates transfers for some unexperienced users,

hindering acceptance.

4. Without stringent auditing and regulatory frameworks, LDT can be exploited or used for money laundering, fraudulent schemes, or terrorism funding.

Our stablecoin was only deployed and tested on Ethereum. However, there is always a possibility for it to be deployed on other chains, like Binance smart chain (BSC), Solana, or Tron blockchain. BSC for example is a hard fork of the Go Ethereum (Geth) protocol, and as such, shares many similarities with the Ethereum blockchain. However, BSC developers have made significant changes in some key areas. The largest change is BSC's consensus mechanism, which allows for cheaper and faster transactions.[18]

Basically, our smart contract code can be implemented on BSC with minor changes.

Supporting cross chain interoperability can prove to have a lot of benefits, including allowing users to choose a more efficient medium of transfer during high congestion times on Ethereum.

Another remark is that requiring users to hold ETH to pay for gas has always been and still is one of the biggest user onboarding challenges, there are two possible solutions to consider for this problem:

1. Implementing the meta transactions feature to the token contract, where a user sends the signed transaction to an operator (paymaster), The operator then takes this signed transaction and submits it to the blockchain paying for the fees himself.

2.Developing an app with a wallet management solution, to be available for unexperienced users and those who can’t obtain ETH to pay gas fees.

REFERENCES

[1] Ali F.Kaeib, Osama A S. Abourodes, Implementation Of Data Encryption Standard On Field Programming Gate Arrays. LICEET2018, (2018).

[2] Teck-Lee Wong, Wee-Yeap Lau, Cashless Payments and Economic Growth: Evidence from Selected OECD Countries July 2020

[3] Rawad Sulieman, Mahmoud Abdalla, Fathi Hamhoum. Electronic payment methods in Libya amidst reality and aspirations. University of Zawiya (2020).
[4] Dr. Gavin Wood. "ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER", Istanbul verion 80085f7, Yellow paper. Available at: https://ethereum.github.io/yellowpaper/paper.pdf. (2021).
[5] Andreas M. Antonopoulos, Gavin Wood, "Mastering Ethereum: Building Smart Contracts and DApps", *O'Reilly Media, Inc.*, 1st edition, (2018).
[6] The ERC-20 token standard, Ethereum foundation, available at: https://ethereum.org/en/developers/docs/standards/tokens/erc-20/ (2021).
[7] Open Zeppelin developers. "Safemath Library". OpenZeppelin documentations.Available https://docs.openzeppelin.com/contracts/2.x/api/math (2021).
[8] Hash functions, NIST, available at: https://csrc.nist.gov/projects/hash-functions (2021).
[9] Shafaq Naheed Khan, Faiza Loukil Blockchain smart contracts: Applications, challenges, and future trends Available at : https://link.springer.com/article/10.1007/s12083-021-01127-0 (2021)
[10] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," Available at: https://bitcoin.org/bitcoin.pdf. (2021)
[11] Christian Catalini, Jai Massari "stablecoins and the future of money", Harvard Business Review, (2021). Available at: https://hbr.org/2021/08/stablecoins-and-the-future-of-money.
[12] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: a technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials, vol. 18, no. 3, pp. 2084-2123,* (2016).
[13] Ethereum Community, "A next-generation smart contract and decentralized application platform," White Paper, available at: https://github.com/ethereum/wiki/wiki/White-Paper. (2021).