A NOVEL CRYPTOCURRENCY - ERC 20 IMPLEMENTATION ON BLOCK CHAIN

Tadepalli Vishal[1] and Mrs P.Kanaka Tulasi[2]

1. M.Tech., BVC college of Engineering, Palacharla ,Rajahmundry.
Asst. Professor Dept of CSE, BVC college of Engineering, Palacharla ,Rajahmundry.

Abstract: *ERC stands for 'Ethereum Request for Comments' and this is an official protocol for proposing improvements to the Ethereum network. The '20' is the unique proposal ID number. ERC20 defines a set of rules which need to be met in order for a token to be accepted and called as an ERC20 Token. So, these particular tokens empower developers of all types to accurately predict how new tokens will function within the larger Ethereum system since these rules are required to interact with each other on the Ethereum Network. According to etherscan.io, the Ethereum Block explorer, there are a total of 37453 Erc20 Token Contracts in existence now and this number is increasing by the minute. It means that there are more than 37,000 tokens now which are piggybacking on the Ethereum Network, hosted by Ethereum addresses and sent by Ethereum Transactions. Since the ERC20 tokens are built atop Ethereum, they are programmable tokens. The potential of their customization is endless. The use cases of ERC20 tokens are significant and varied according to the token's needs. The Dock token also has a multitude of applications built on the Ethereum network.*

Introduction:

The ERC20 token standard is a set of criteria developed by the Ethereum community for how to set up smart contracts on the blockchain in order to ensure optimal interoperability. ERC stands for Ethereum Request for Comments and this particular request was to help improve the standards to which developers should strive to adhere in developing smart tokens

The ERC20 Token Standard is a set of 6 functions and 2 events that need to be included (in a specific agreed upon language and format) in a smart token contract in order to enable interoperability across multiple interfaces and distributed applications (dapps). The 6 functions relate to (1) how tokens can be transferred (either by its owner or on behalf of its owner) and (2) how to access data about the token (name, symbol, total supply, and account balance). The 2 events are transfers and approvals, and how they need to be formatted.

ERC20 token standard implementation assumes two ways of token transferring: (1) transfer function and (2) approve + transferFrom pattern.

It should be noted that event handling is a well-known and standard practice in programming. In case of token standard, a token transfer should be considered as an event. transfer function of ERC20 does not provide any opportunity to handle the transfer, i.e. it is silently increasing balance of the receiver. It is impossible for the receiver to recognize that transfer occurs if the receiver is a contract. As the result, the only correct way to make a token deposit to a contract is approve + transfer From pattern.

Expansion of ERC-20 System

There are most widely used authentication procedures in current days.

They are

Connects the User Data across the web

Today, user data powers almost every consumer facing app on the web. In a utopian world, the apps using your data must communicate and interact with each other solving the problem of multiplicity of data and eliminates the process of manual migration of data. So, the Dock.io protocol solves this problem as an open network bringing about a revolution on how apps communicate with each other.

Token based grade systems.

One of the key advantages of an ERC20 token is how it lets the token holders to participate in the voting process. The Dock token holders can introduce new proposals and vote on the future development roadmap of the protocol. The roadmap of any open organisation is a living, breathing document which is not resistant to change. In fact, the future of the protocol lies in the hands of the token holders who have the power to influence and be heard about the direction and the future of the protocol which creates a truly decentralised economy for data exchange on the web.

Accessing, Sharing and ensuring the Quality of Data

By Design, the tokens have a functional necessity in terms of accessing, sharing and ensuring the quality of data.

a) Applications use the Dock tokens when users want to access data on their platform.

b) Applications are rewarded with Dock tokens when users create new data to share via the Dock protocol.

c) The quality of data is also ensured because the applications are only rewarded when other applications accept the data.

With all of these advantages of the ERC20 token and more, they are averse to lesser amounts of risk, increased uniformity, reduced complexity, and enhanced liquidity of the Dock tokens which prompted the core team to invest in the ERC-20 as it is a crucial aspect of Ethereum now and is going to be a big part of how it will be shaped in the future.

Implementation of ERC-20 Token:

Earlier this week the ERC-20 token interface became a formal improvement proposal, freezing the definition. This article takes a look at tokens and explains the features and functions of ERC-20 to provide an understanding of what token contracts are and how developers can work with them.

ERC-20 came about as an attempt to provide a common set of features and interfaces for token contracts in Ethereum, and has proved to be very successful. ERC-20 has many benefits, including allowing wallets to provide token balances for hundreds of different tokens and creating a means for exchanges to list more tokens by providing nothing more than the address of the token's contract. The benefits of creating a token as ERC-20 compliant is such that very few token contracts today are created any other way.

There remains a lot of confusion around what token contracts really are. Essentially, a token contract is a smart contract that contains a map of account addresses and their balances. The balance represents a value that is defined by the contract creator: one token contract might use balances to represent physical objects, another monetary value,

and a third the holder's reputation. The unit of this balance is commonly called a token.

Here are a few basic terms we are going to use in this article. If you are familiar with the following concepts, feel free to skip to the next section.

- **Ethereum based ERC20 Tokens:** In Ethereum tokens represent any tradable goods such as coins, loyalty points etc. You can create your own crypto-currencies based on Ethereum. Additionally the benefit of following ERC20 standard is that your tokens will be compatible with any other client or wallets that use the same standards.

- **Smart Contracts:** Smart Contracts are self executing code blocks deployed on the Ethereum blockchain. They contain data & code functions. Contracts make decisions, interact with other contracts, store data and transfer Ether (the unit of crypto-currency in the Ethereum blockchain) among users.

- **Solidity:** A language for writing smart contracts.

- **MetaMask/Mist/MEW Wallet:** A digital facility that holds your Ether and other Ethereum based tokens.

Standard of Ethereum's ERC20 Token

The Ethereum blockchain distinguishes itself from other blockchains with its innovative decentralized application (DApp) functionality. Using tokens, developers can build and launch their very own DApp on the Ethereum blockchain. However, the opportunity to create a wide range of token based DApps also comes with problems, and the implementation of ERC20 (Ethereum Request for Comments 20) is designed to tackle those issues.

Firstly, a token is nothing more than a smart contract that runs on top of the Ethereum blockchain, with the behavior of the token being dictated by its code base. Tokens are typically incorporated into DApps to represent anything of value, from digital assets, to objects in the physical world. Before ERC20, if a developer wanted to enable the cross trading of tokens, e.g. trading 'Token A' for 'Token B', the developer would have to closely examine the intricate code base behind Token A and Token B to handle the trade. This process soon becomes an arduous and complex one if a developer wants to implement the cross trading of Token A with Tokens B – Z. The code base behind each token would have to studied in order to implement a simple token exchange. ERC20 is designed to tackle this problem by establishing a common set of rules that developers must adhere to in order for their token to be ERC20 compliant. Although not an obligatory standard, developers are encouraged to adhere to the ERC20 standards because of the benefits it brings. As well as easier cross-token exchange, ERC20 compliant tokens can also seamlessly interact with various wallets and exchanges because developers of wallets and exchanges already understand how compliant tokens will behave.

In order for a token to be ERC20 compliant, the code base behind the token must be able to perform the following functions:

- Retrieve the total token supply.

- Retrieve the token holder's account balance.

- Transfer the token from the owner to another party.

- Approve use of the token as a monetary asset.

ERC20 is a major step for the Ethereum blockchain. With aims to create an ecosystem of seamlessly integrated DApps, ERC20 is a positive step in the right direction by encouraging interoperability between tokens.

There are two different ways to transfer ERC20 tokens depending on whether you intend to send the tokens directly or delegate the transfer to another smart contract. You can either call transfer to send tokens to a wallet address or call approve, and then trigger transferfrom the receiver contract, in order for it to be aware of the transfer and handle it accordingly.

The token fallback function, which will be called at the receiver contract, must be named tokenFallback and take the parameters:(address, uint256, bytes). It's an analogue of the fallback function for ETH transactions and should be used to handle incoming transactions.

Inability of handling incoming token transactions

By sending ERC20 tokens using the transfer function, the token contract is not notifying the receiver that a transaction had occurred. The tokens are just simply credited to the address of the receiver. In addition to that, there is no way to handle incoming token transactions on contracts and no way to reject or handle any non-supported tokens.

In addition to preventing tokens getting lost, the new transfer method will also allow the smart contract to actively handle sent tokens (e.g., an exchange smart contract can react to a token transfer by crediting the token balance of the user

Token Transfer Uniformity

An ERC20 token transaction between a regular/non-contract address and contract are two different transactions: You should call approve on the token contract and then call transferFrom on the other contract when you want to deposit your tokens into it.

ERC-20 simplifies this requirement and allows using the same transfer function. ERC-20 tokens can be sent by calling transfer function on the token contract with no difference if the receiver is a contract or a wallet address, since there is a new way to notify the receive contract of the transfer.

If the receiver is a a regular/non-contract address, an ERC-20 token transfer will be the same as an ERC-20 transfer. On the other hand, if the receiver is a contract, then the ERC-20 token contract will try to call tokenFallback function on receiver contract. If there is no tokenFallback function on receiver contract, the transaction will fail.

For example, a decentralized exchange will no longer need to force users to call approve at the token contract, then call deposit to call transferFrom and take allowed tokens. The token transaction will automatically be handled inside the exchange contract, via the tokenFallback function.

ERC-20 Empowers Developers

In short, the ERC-20 defines a common list of rules for all Ethereum tokens to follow, meaning that this particular token empowers developers of all types to accurately predict how new tokens will function within the larger Ethereum system. The impact that ERC-20 therefore has on developers is massive, as projects do not need to be redone each time a new token is released. Rather, they are designed to be compatible with new tokens, provided those tokens

adhere to the rules. Developers of new tokens have by-and-large observed the ERC-20 rules, meaning that most of the tokens released through Ethereum initial coin offerings are ERC-20 compliant.

ERC-20 Specifies Six Functions

ERC-20 defines six different functions for the benefit of other tokens within the Ethereum system. These are generally basic functionality issues, including how tokens are transferred and how users can access data about a token. ERC-20 also prescribes two different signals that each token takes on and which other tokens are attuned to.

Put together, this set of functions and signals ensures that Ethereum tokens of different types will typically work the same in any place within the Ethereum system. This means that almost all of the wallets which support the ether currency also support ERC-20 compliant tokens.

ERC-20 is technically still in draft form, meaning that it has gone unenforced by the broader Ethereum community. Still, it seems that the momentum is strong enough that all new tokens are highly likely to conform to the ERC-20 rules. Because the standard remains young, there will likely be some troubleshooting which must occur as Ethereum continues to develop. One significant issue with Ethereum tokens so far is that tokens sent directly to a smart contract will lose money. An error in the protocol means that a token's contract cannot respond to an attempt to make a direct transfer, resulting in the "loss" of the money associated with that transfer.

Process Authentic Functioning of ERC:

ERC stands for Ethereum Request for Comments. An ERC is authored by Ethereum community developers in the form of a memorandum describing methods, behaviors, research, or innovations applicable to the working of the Ethereum ecosystem. It is submitted either for peer review or simply to convey new concepts or information. After core developers and community approval, the proposal becomes a standard.

Therefore, as a result, we have a set of standards or proposals (e.g. for tokens). Actually, these rules are simple set of functions that Smart Contract should implement. In return, contracts, implementing the standard can be used via a single interface. The best example is ERC-20 standard. All Smart Contracts implementing this standard, by default can be listed to crypto exchanges without any extra technical work.

ERC-20

It is the most common and well-known standard within all crypto community. 99% (if not all) issued ICO tokens on top of the Ethereum implements this standard. Actually, it is just a simple set of functions that your token code has to have. For those who can read the code, the contract below is very simple to understand.

The key benefit we get here, is that any application or other smart contract can interact with a token in a standard manner without a need of knowing other details about the token.

Therefore, we have a very pleasant way to create any ICO token and have a standard way to interact with all of them like they are all the same. For instance, crypto wallet developers can avoid custom development and integrations to add new tokens. All they need to know is the Ethereum Token address that implements the standard.

Enhance of ERC-20

There is no phrase like "If you send your tokens to the address or a contract that is not intended to work with tokens then your tokens are permanently lost." We should keep in mind that we are talking about the Ethereum smart-contracts. In case of Ether transaction, every time a transaction is invoked incorrectly (a receiver is unable to handle transaction or a receiver is a contract that is not intended to work with Ether), an error is thrown and the transaction is reverted.

Thus, the expected behavior is that in the event of an error caused by the recipient's inability to handle the incoming transaction, such a transaction should fail.

Ethereum users are often familiar with Ether transactions. As the result, the expected behavior is that if a transfer of tokens is invoked incorrectly (a receiver is unable to handle transaction or a receiver is a contract that is not intended to work with tokens), an error must be thrown.

The statement "... produce an incorrect or unexpected result, or to behave in unintended ways", which is a property of a program error, is of decisive importance here. The "bug of ERC20 transfers" is the unexpected behavior of transfer function in this case.

The statement "... prompting a user to make a security decision without giving the user enough information to answer it. " which is a property of Software Vulnerability, is of decisive importance here.

There is no sufficient information at the ERC20 token standard definition about the behavior of transfer function in case of the error-handling. A phrase "WARNING: If you will call this function to transfer your tokens to any contract then your tokens are permanently lost." MUST be added to the definition of transfer function of the ERC20 token standard.

A phrase "Calling the transfer function to transfer your tokens to contracts is prohibited in this token standard." MUST be added to the Abstract section of the ERC #20 token standard.

This is required to make each token and UI developer aware of what they are working with. This is what is missing at ERC20 standard.

As soon as this will be implemented, each token developer and UI developer MUST place a big red banner "WARNIGN: You are attempting to use ERC20 token, you should know that if you call a transfer function to send tokens to a contract then your tokens are permanently lost." Otherwise it will be a Software Vulnerability: User Interface fault

Authors:

1. TADEPALLI VISHAL, M.Tech., BVC college of Engineering, Palacharla, Rajahmundry. (146M1D5804)

2. Mrs P. KANAKA TULASI is working as Associate professor in Department of Computer Science & Engineering at BVC College of Engineering, Rajahmundry. She has a total technical experience of 7 years.

References:

1. S. Chiasson, P. van Oorschot, and R. Biddle, “Graphical Password Authentication Using Cued Click Points,” Proc. European Symp.Research in Computer Security (ESORICS), pp. 359-374, Sept. 2007

2. Anderson, F. Bergadano, B. Crispo, J.H. Lee, C. Manifavas, R. Needham, “A New Family of Authentication Protocols”, ACM OSR, 1998.

3. B. Groza, "Broadcast authentication protocol with time synchronization and quadratic residues chains", Second International Conference on Availability, Reliability and Security (ARES’07), pp. 550-557, IEEE Comp. Soc., 2007.

4. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, “SPINS: Security Protocols for Sensor Network”, Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM, 2001.

5. Groza, T.L. Dragomir, "On the use of one-way chain based authentication in secure control systems", Second International Conference on Availability, Reliability and Security ARES’07), pp. 1214-1221, IEEE Comp. Soc., 2007.

6. Authentication Mechanism” IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 9, NO. 2, MARCH/APRIL 2012

7. FIPS 180-1, National Institute of Standards and Technology (NIST). “Announcing the Secure Hash Standard”, U.S. Department of Commerce, 1995.

8. Groza, “Using one-way chains to provide message authentication without shared secrets”, Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, SecPerU 2006, IEEE Comp. Soc., 2006.

9. Mitra Sing, D. Cavagnino, D.Narasimha, “Individual Authentication in Multiparty Communications”. Computer & Security, Elsevier Science, vol. 21 n. 8, 2002, pp.719-735.

10. N. Haller, C. Metz, P. Nesser, M. Straw, “A One-Time Password System”, RFC 2289, Bellcore, Kaman Sciences Corporation, Nesser and Nesser Consulting, 1998.

11. L. Lamport, “Password Authentication with Insecure Communication”, Communication of the ACM, 24, 770-772, 1981.