2023/2024 S2

Ethereum III

Dr. Hui Gong (h.gong@westminster.ac.uk)

UNIVERSITY OF WESTMINSTER

Tokens

How Tokens are Used

The most obvious use of tokens is as digital private currencies. However, this is only one possible use. Tokens can be programmed to serve many different functions, often overlapping. For example, a token can simultaneously convey a voting right, an access right, and ownership of a resource. As the following list shows, currency is just the first “app”:

Currency
A token can serve as a form of currency, with a value determined through private trade.

Resource
A token can represent a resource earned or produced in a sharing economy or resource-sharing environment; for example, a storage or CPU token representing resources that can be shared over a network.

Asset
A token can represent ownership of an intrinsic or extrinsic, tangible or intangible asset; for example, gold, real estate, a car, oil, energy, MMOG items, etc.

Access
A token can represent access rights and grant access to a digital or physical property, such as a discussion forum, an exclusive website, a hotel room, or a rental car.

Equity
A token can represent shareholder equity in a digital organization (e.g., a DAO) or legal entity (e.g., a corporation).

Voting
A token can represent voting rights in a digital or legal system.

Collectible
A token can represent a digital collectible (e.g., CryptoPunks) or physical collectible (e.g., a painting).

Identity
A token can represent a digital identity (e.g., avatar) or legal identity (e.g., national ID).

Attestation
A token can represent a certification or attestation of fact by some authority or by a decentralized reputation system (e.g., marriage record, birth certificate, college degree).

Utility
A token can be used to access or pay for a service.

Tokens and Fungibility

“In economics, fungibility is the property of a good or a commodity whose individual units are essentially interchangeable.”

Tokens are fungible when we can substitute any single unit of the token for another without any difference in its value or function.

Strictly speaking, if a token’s historical provenance can be tracked, then it is not entirely fungible. The ability to track provenance can lead to blacklisting and whitelisting, reducing or eliminating fungibility.

Non-fungible tokens are tokens that each represent a unique tangible or intangible item and therefore are not interchangeable. For example, a token that represents ownership of a specific Van Gogh painting is not equivalent to another token that represents a Picasso, even though they might be part of the same “art ownership token” system. Similarly, a token representing a specific digital collectible such as a specific CryptoKitty is not interchangeable with any other CryptoKitty. Each non-fungible token is associated with a unique identifier, such as a serial number.

ERC721: Non-fungible Token (Deed) Standard

Counterparty risk is the risk that the other party in a transaction will fail to meet their obligations. Some types of transactions suffer additional counterparty risk because there are more than two parties involved. For example, if you hold a certificate of deposit for a precious metal and you sell that to someone, there are at least three parties in that transaction: the seller, the buyer, and the custodian of the precious metal. Someone holds the physical asset; by necessity they become party to the fulfillment of the transaction and add counterparty risk to any transaction involving that asset.

Some tokens represent digital items that are **intrinsic** to the blockchain. Those digital assets are governed by consensus rules, just like the tokens themselves. This has an important implication: tokens that represent intrinsic assets do not carry additional counterparty risk. If you hold the keys for a CryptoKitty, there is no other party holding that CryptoKitty for you - you own it directly. The blockchain consensus rules apply and your ownership (i.e., control) of the private keys is equivalent to ownership of the asset, without any intermediary.

Using Tokens: Utility or Equity

Almost all projects in Ethereum today launch with some kind of token. But do all these projects really need tokens? Are there any disadvantages to using a token, or will we see the slogan “tokenize all the things” come to fruition? In principle, the use of tokens can be seen as the ultimate management or organization tool.

Let’s start by clarifying the role of a token in a new project. The majority of projects are using tokens in one of two ways: either as “utility tokens” or as “equity tokens.” Very often, those two roles are conflated.

Utility tokens are those where the use of the token is required to gain access to a service, application, or resource. Examples of utility tokens include tokens that represent resources such as shared storage, or access to services such as social media networks.

Equity tokens are those that represent shares in the control or ownership of something, such as a startup. Equity tokens can be as limited as nonvoting shares for distribution of dividends and profits, or as expansive as voting shares in a decentralized autonomous organization, where management of the platform is through some complex governance system based on votes by the token holders.

The real problem is that utility tokens introduce significant risks and adoption barriers for startups. Perhaps in a distant future “tokenize all the things” will become reality, but at present the set of people who have an understanding of and desire to use a token is a subset of the already small cryptocurrency market.

For a startup, each innovation represents a risk and a market filter. Innovation is taking the road least traveled, walking away from the path of tradition. It is already a lonely walk. If a startup is trying to innovate in a new area of technology, such as storage sharing over P2P networks, that is a lonely enough path. Adding a utility token to that innovation and requiring users to adopt tokens in order to use the service compounds the risk and increases the barriers to adoption. It’s walking off the already lonely trail of P2P storage innovation and into the wilderness.

Advocates of “tokenize all the things” will likely counter that by adopting tokens they are also inheriting the market enthusiasm, early adopters, technology, innovation, and liquidity of the entire token economy. That is true too. The question is whether the benefits and enthusiasm outweigh the risks and uncertainties.

https://www.finma.ch/en/news/2018/02/20180216-mm-ico-wegleitung/

Tokens on Ethereum

Blockchain tokens existed before Ethereum. In some ways, the first blockchain currency, Bitcoin, is a token itself. Many token platforms were also developed on Bitcoin and other cryptocurrencies before Ethereum. However, the introduction of the first token standard on Ethereum led to an explosion of tokens.

Before we delve into the details of creating tokens on Ethereum, it is important to have an overview of how tokens work on Ethereum. Tokens are different from ether because the Ethereum protocol does not know anything about them. Sending ether is an intrinsic action of the Ethereum platform, but sending or even owning tokens is not. The ether balance of Ethereum accounts is handled at the protocol level, whereas the token balance of Ethereum accounts is handled at the smart contract level. In order to create a new token on Ethereum, you must create a new smart contract. Once deployed, the smart contract handles everything, including ownership, transfers, and access rights. You can write your smart contract to perform all the necessary actions any way you want, but it is probably wisest to follow an existing standard.

The first standard was introduced in November 2015 by Fabian Vogelsteller as an Ethereum Request for Comments (ERC). It was automatically assigned GitHub issue number 20, giving rise to the name “ERC20 token.” The vast majority of tokens are currently based on the ERC20 standard. The ERC20 request for comments eventually became **Ethereum Improvement Proposal 20 (EIP-20)**, but it is mostly still referred to by the original name, ERC20.

ERC20 is a standard for fungible tokens, meaning that different units of an ERC20 token are interchangeable and have no unique properties.

The ERC20 standard defines a common interface for contracts implementing a token, such that any compatible token can be accessed and used in the same way. The interface consists of a number of functions that must be present in every implementation of the standard, as well as some optional functions and attributes that may be added by developers.

The ERC20 Token Standard

ERC20 required functions and events

An ERC20-compliant token contract must provide at least the following functions and events:

`totalSupply`
Returns the total units of this token that currently exist. ERC20 tokens can have a fixed or a variable supply.

`balanceOf`
Given an address, returns the token balance of that address.

`transfer`
Given an address and amount, transfers that amount of tokens to that address, from the balance of the address that executed the transfer.

`transferFrom`
Given a sender, recipient, and amount, transfers tokens from one account to another. Used in combination with approve.

`approve`
Given a recipient address and amount, authorizes that address to execute several transfers up to that amount, from the account that issued the approval.

`allowance`
Given an owner address and a spender address, returns the remaining amount that the spender is approved to withdraw from the owner.

`Transfer`
Event triggered upon a successful transfer (call to transfer or transferFrom) (even for zero-value transfers).

`Approval`
Event logged upon a successful call to approve.

ERC20 optional functions

In addition to the required functions listed in the previous section, the following optional functions are also defined by the standard:

`name`
Returns the human-readable name (e.g., “US Dollars”) of the token.

`symbol`
Returns a human-readable symbol (e.g., “USD”) for the token.

`decimals`
Returns the number of decimals used to divide token amounts.

The ERC20 Token Standard

The ERC20 interface defined in Solidity

Here's what an ERC20 interface specification looks like in Solidity:

```
contract ERC20 {
   function totalSupply() constant returns (uint theTotalSupply);
   function balanceOf(address _owner) constant returns (uint balance);
   function transfer(address _to, uint _value) returns (bool success);
   function transferFrom(address _from, address _to, uint _value) returns
      (bool success);
   function approve(address _spender, uint _value) returns (bool success);
   function allowance(address _owner, address _spender) constant returns
      (uint remaining);
   event Transfer(address indexed _from, address indexed _to, uint _value);
   event Approval(address indexed _owner, address indexed _spender, uint _value);
}
```

ERC20 data structures

If you examine any ERC20 implementation you will see that it contains two data structures, one to track balances and one to track allowances. In Solidity, they are implemented with a data mapping.

The first data mapping implements an internal table of token balances, by owner. This allows the token contract to keep track of who owns the tokens. Each transfer is a deduction from one balance and an addition to another balance:

```
mapping(address => uint256) balances;
```

The second data structure is a data mapping of allowances. As we will see in the next section, with ERC20 tokens an owner of a token can delegate authority to a spender, allowing them to spend a specific amount (allowance) from the owner's balance. The ERC20 contract keeps track of the allowances with a two-dimensional mapping, with the primary key being the address of the token owner, mapping to a spender address and an allowance amount:

```
mapping (address => mapping (address =>
uint256)) public allowed;
```

ERC20 workflows: “transfer” and “approve & transferFrom”

The ERC20 token standard has two transfer functions. You might be wondering why. ERC20 allows for two different workflows. The first is a single-transaction, straightforward workflow using the transfer function. This workflow is the one used by wallets to send tokens to other wallets. The vast majority of token transactions happen with the transfer workflow.

Executing the transfer contract is very simple. If Alice wants to send 10 tokens to Bob, her wallet sends a transaction to the token contract’s address, calling the transfer function with Bob’s address and 10 as the arguments. The token contract adjusts Alice’s balance (–10) and Bob’s balance (+10) and issues a Transfer event.

The second workflow is a two-transaction workflow that uses approve followed by transferFrom. This workflow allows a token owner to delegate their control to another address. It is most often used to delegate control to a contract for distribution of tokens, but it can also be used by exchanges.

If a company is selling tokens for an ICO, they can approve a crowdsale contract address to distribute a certain amount of tokens. The crowdsale contract can then transferFrom the token contract owner’s balance to each buyer of the token, as illustrated in Figure.

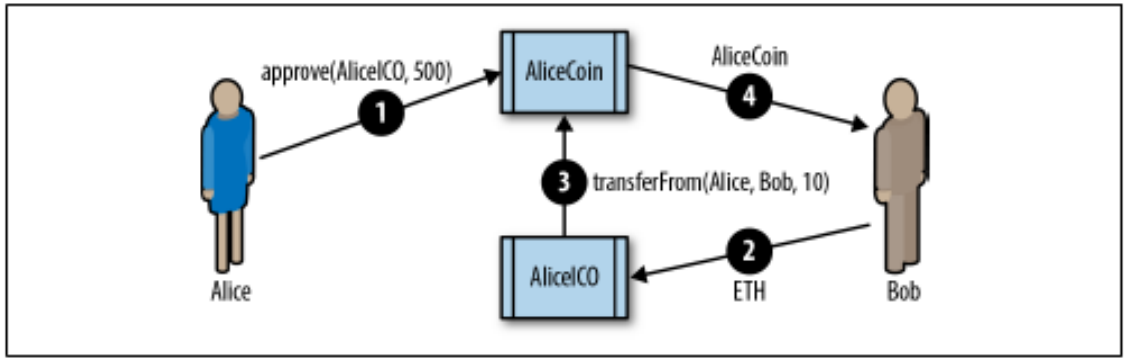

Figure 10-1. The two-step approve & transferFrom workflow of ERC20 tokens

An ***Initial Coin Offering (ICO)*** is a crowdfunding mechanism used by companies and organizations to raise money by selling tokens. The term is derived from Initial Public Offering (IPO), which is the process by which a public company offers shares for sale to investors on a stock exchange. Unlike the highly regulated IPO markets, ICOs are open, global, and messy. The examples and explanations of ICOs in this book are not an endorsement of this type of fundraising.

Launching Our Own ERC20 Token

References to read

[1] https://github.com/ConsenSys/Tokens/blob/master/contracts/eip20/EIP20.sol

[2] https://github.com/ConsenSys

[3] https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v1.12.0/contracts/token/ERC20/StandardToken.sol

[4] https://github.com/OpenZeppelin

[5] https://docs.openzeppelin.com/contracts/3.x/

[6] Deloitte, 2018, The tokenization of assets disrupting financial industry

[7] Deloitte, 2020, Are token assets the securities tomorrow?

[8] Deloitte, 2020, Tokenization – the future of the platform business model

[9] EY, 2020, Tokenization of Assets