



以太坊代币

北京交通大学
计算机与信息技术学院
信息安全系

李超 (li.chao@bjtu.edu.cn)
段莉 (duanli@bjtu.edu.cn)

以太坊上的代币

- 在以太坊上的众多智能合约之中，应用最为广泛的是**代币合约（Token Contract）**
- 代币合约是在以太坊上管理账户及其拥有的代币的**智能合约**，实质上可以理解为一张账户地址和对应账户代币余额的**映射表**
- 以太坊上的代币可以被称为**数字资产**，记录资产数据的代币合约就是一份**账本**

以太坊上的代币

- 一般意义上的加密货币，如比特币、以太坊和莱特币等，是记录在账户状态中，**直接存储在区块内的数据**，伴随“挖矿”等机制发行，通过交易的方式流通
- 代币以以太坊区块链为平台，**记录在更高层的代币合约中**，具体来讲是存储在以太坊交易消息数据字段的可执行代码中

以太坊上的代币

- 代币**无需“挖矿”**，其创建者可以通过智能合约定义自己的代币发行标准，直接在合约代码中实现**“铸币”**功能
- 代币的**流通**是通过在以太坊**交易中调用智能合约的函数接口进行转账**，代币合约创建者同样可以在这一过程中添加一些自定义的操作
- 代币拥有更高的**灵活性**，并且其**安全性**也由以太坊区块链机制和智能合约代码保证

Token Tracker

ERC-20













ERC-20 Tokens

A total of 284,286 Token Contracts found

First

Page 1 of 20

Last

#	Token	Price	Change (%)	Volume (24H)	Market Cap	Holders
1	 Tether USD (USDT) Digital money for a digital age.	\$1.0029 0.0000847176 Btc 0.002426 Eth	 0.05%	\$43,740,468,975	\$10,027,508,173	1,875,844
2	 ChainLink Token (LINK) A blockchain-based middleware, acting as a bridge between cryptocurrency smart contracts, data feeds, APIs and traditional bank account payments.	\$16.3431 0.0013805025 Btc 0.039531 Eth	 -0.49%	\$2,876,004,631	\$5,720,074,361	209,300
3	 BNB (BNB) Binance aims to build a world-class crypto exchange, powering the future of crypto finance.	\$22.6788 0.0019156811 Btc 0.054857 Eth	 -2.90%	\$265,468,225	\$3,274,964,341	313,786
4	 Bitfinex LEO Token (LEO) A utility token designed to empower the Bitfinex community and provide utility for those seeking to maximize the output and capabilities of the Bitfinex trading platform.	\$1.2642 0.0001067914 Btc 0.003058 Eth	 -0.38%	\$13,574,055	\$1,263,615,732	1,841
5	 USD Coin (USDC) USDC is a fully collateralized US Dollar stablecoin developed by CENTRE, the open source project with Circle being the first of several forthcoming issuers.	\$1.0021 0.0000846492 Btc 0.002424 Eth	 0.11%	\$427,470,045	\$1,257,439,466	228,142
6	 VeChain (VEN) Aims to connect blockchain technology to the real world by as well as advanced IoT integration.	\$0.0184 0.0000015535 Btc 0.000044 Eth	 -6.33%	\$193,360,803	\$1,019,875,747	45,732

代币的发行与流通

- **铸造代币**：当其他账户通过向合约转入以太币或以其他方式**调用合约铸造功能**时，该代币合约向账户对应的余额值增加相应数量的代币，代币的总供应盘也相应增加，完成铸币
 - 例如，账户 **Charlie** 调用代币合约的铸币功能函数，合约经验证后在将其余额增加 **50** 个代币，同时代币总供应量也增加 **50** 个代币

代币的发行与流通

- **销毁代币**：账户通过调用合约的**销毁功能函数**，销毁其账户余额中的代币，代币总供应量也相应地减少
 - 通常代币合约的代币销毁功能是通过向特殊的**零地址 0x000... 0000** 中转入相应数量的代币来完成，此时代币总供应量不会减少

代币的发行与流通

- **代币转账**：代币转账是代币合约的一项基本功能，也是数字资产流通功能的具体实现
 - 例如，账户 **Alice** 调用合约的转账功能函数，向账户 **Bob** 转入 50 个代币，此时合约中记录的 **Alice** 账户余额减少了 50，而 **Bob** 账户余额增加了 50
- 此外，代币合约还可以加入数字资产的查询、权限控制，甚至经济学公式计算等功能

ERC20 – 标准定义

- **ERC 20 代币合约标准**规定了一个以太坊代币合约所需实现的函数功能和事件记录
- 该标准满足了代币作为数字资产所必须具备的一些**基本功能和要求**，如注明代币名称、代币转账、本账户中允许链上第三方使用的代币限额等
- **ERC 20** 的出现为以太坊上的代币合约提供了一个标准化的方案，也对以太坊上数字资产的实现进行了一定的规范

ERC20 – 标准接口

```
contract ERC20 {  
    string public constant name = "Token Name";  
    string public constant symbol = "SYM";  
    uint8 public constant decimals = 18;  
    function totalSupply() constant returns (uint supply);  
    function balanceOf( address who ) constant returns (uint value);  
    function allowance( address owner, address spender ) constant returns (uint allowance);  
    function transfer( address to, uint value ) returns (bool ok);  
    function transferFrom( address from, address to, uint value ) returns (bool ok);  
    function approve( address spender, uint value ) returns (bool ok);  
    event Transfer( address indexed from, address indexed to, uint value);  
    event Approval( address indexed owner, address indexed spender, uint value);  
}
```

ERC20 – 标准接口

代币名称 (name)

```
string public constant name = "Token Name";
```

- 由代币合约创建者指定的完整名称，是一串公开的字符串常量，如 **LiChaoToken**
- 无法保证一个代币名称唯一标识一种特定的合约代币
- 交易所通过注册机制可以检查并保证代币名称与代币合约一一对应

ERC20 – 标准接口

代币符号 (symbol)

```
string public constant symbol = "SYM";
```

- 由代币合约创建者指定的代币简称，是一串公开的字符串常量，一般由 3 ~ 4 个大写字母组成，便于标识该代币，如 LCT
- 符合 ERC 20标准的代币可以通过在各交易所中注册，使其代币符号能够唯一标识该代币合约

ERC20 – 标准接口

小数点位（decimals）

```
uint8 public constant decimals = 18;
```

- 由代币合约创建者指定的一个公开无符号整数常量，用于指定代币的**最小精度值**，一般为 18
- 小数点位的数值表示该代币在交易中最小单位在小数点后的位数
- 设置原因是以太坊虚拟机不支持小数计算

ERC20 – 标准接口

总供应量 `totalSupply()`

```
function totalSupply() constant returns (uint supply);
```

- 用于查看代币当前的总供应量，即当前合约账本中所有账户余额的总和
- 该函数没有输入参数，返回值为无符号整数常量

ERC20 – 标准接口

余额 `balanceOf()`

```
function balanceOf( address who ) constant returns (uint value);
```

- 用于查看当前合约中**指定账户的代币余额**
- 该函数的输入参数为 账户地址，返回值为 账户代币余额，为无符号整数常量

ERC20 – 标准接口

转账 `transfer()`

```
function transfer( address to, uint value) returns (bool ok);
```

- 用于从当前账户向其他账户进行代币转账
- 输入参数为目标账户地址和转账的代币数额，返回值为布尔型变量
- 当账户**满足**当前有足够的余额、转账数额为正数以及合约编写者指定的其他**条件**时，转账成功，函数返回值为真；否则转账失败，函数返回值为假

ERC20 – 标准接口

从他人处转账 **transferFrom()**

```
function transferFrom( address from, address to, uint value) returns (bool ok);
```

- 用于从他人账户向其他账户进行代币转账
- 用户不仅可以自己使用 **transfer()** 函数自己发起转账，还可以**授权他人**在一定限额下调用 **transferFrom()** 函数从自己账户中转出代币，而无须自己介入
 - 例如，银行合约，自动完成转账过程，而无须通知用户参与

ERC20 – 标准接口

允许量值 `approve()`

```
function approve( address spender, uint value ) returns (bool ok);
```

- ERC 20 标准引入了允许量值 `allowed[A][B]` 记录的是用户 A 对本账户中允许账户 B 转走的代币额度
- 当账户 B 调用 `transferFrom()` 函数从账户 A 中转出代币时，需先通过检查，确保转出的数额不超过账户 A 设置的 `allowed[A][B]` 值，并且转账之后 `allowed[A][B]` 值会减少相应的数额

ERC20 – 标准接口

限额 `allowance()`

```
function allowance( address owner, address spender ) constant returns (uint allowance);
```

- 用于查看当前的 `allowed` 值
- 该函数的输入参数为代币持有方 **A** 的地址和代币使用方 **B** 的地址，返回值为当前在账户 **A** 中允许账户 **B** 转出的代币数额 `allowed[A][B]`，为无符号整型常量

ERC20 – 标准接口

事件：转账

```
event Transfer( address indexed from, address indexed to, uint value);
```

- 用于记录代币合约最基本的功能，转账
- 事件的输入参数为转账的发起方、接收方以及转账的代币金额，一般位于 **transfer函数**和 **transferFrom函数**中转账成功之后触发
- 用户可以从交易收据（**receipt**）中查看每一笔代币转账的相关信息

ERC20 – 标准接口

事件：允许

```
event Approval( address indexed owner, address indexed spender, uint value);
```

- 用于记录代币合约的进阶功能，允许他人从本账户中转出代币
- 事件的输入是代币的持有者、使用者以及所设置的允许金额，一般位于 **approve函数** 中，设置允许限额成功之后触发
- 用户可以从交易收据（**receipt**）中查看代币持有者对他人设置的允许转账限额等相关信息

ERC20 – 以EOS为例

- EOS 代币是由 Daniel Larime 等人开发的 EOS.IO 项目所发行的一种代币
- EOS.IO 是一款新一代的区块链项目，针对以太坊现有的一些问题作出改进。采用股权委托证明（**DPoS**）的共识算法，提高吞吐量 and 用户数量，可支持每秒百万级别的交易量；将交易延迟降低至数秒，并且拥有冻结功能，更便于项目升级和问题修复

ERC20代币

```
function transfer(address dst, uint wad) returns (bool) {  
    assert(_balances[msg.sender] >= wad);  
  
    _balances[msg.sender] = sub(_balances[msg.sender], wad);  
    _balances[dst] = add(_balances[dst], wad);  
  
    Transfer(msg.sender, dst, wad);  
  
    return true;  
}
```

- 函数首先判断转账发送方，即转账消息的发起方 `msg.sender` 的余额是否足够
- 然后发送方 `msg.sender` 余额减少，接收方 `dst` 余额增加
- 再触发 **Transfer** 事件记录转账的 `msg.sender`、`dst` 和 `wad`
- 最后，转账成功，返回真值

ERC20代币

```
function transferFrom(address src, address dst, uint wad) returns (bool) {  
    assert(_balances[src] >= wad);  
    assert(_approvals[src][msg.sender] >= wad);  
  
    _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);  
    _balances[src] = sub(_balances[src], wad);  
    _balances[dst] = add(_balances[dst], wad);  
  
    Transfer(src, dst, wad);  
  
    return true;  
}
```

- 除了判断发送方余额是否足够之外，还需判断发送方 **src** 对转账发起者 **msg.sender** 的**允许限额** **_approvals** 是否足够
- 转账过程中，除了发送方、接收方的余额变更之外，还需将发送方对转账发起者的**允许限额** **_approvals[src][msg.sender]**扣除相应转账数额
- 同样触发 **Transfer** 事件进行记录
- 转账成功，返回真值

ERC20代币

```
function approve(address guy, uint256 wad) returns (bool) {  
    _approvals[msg.sender][guy] = wad;  
  
    Approval(msg.sender, guy, wad);  
  
    return true;  
}
```

- 将调用者 `msg.sender` 对被授权方 `guy` 的允许值 `_approvals` 进行更改
- 触发了 `Approval` 事件来记录授权方 `msg.sender`、被授权方 `guy` 和允许限额 `wad`，并返回真值

ERC20代币

```
function mint(uint128 wad) auth stoppable note {  
    _balances[msg.sender] = add(_balances[msg.sender], wad);  
    _supply = add(_supply, wad);  
}  
function burn(uint128 wad) auth stoppable note {  
    _balances[msg.sender] = sub(_balances[msg.sender], wad);  
    _supply = sub(_supply, wad);  
}
```

- 两个函数均带有 **auth 函数修改器**，限定只能由**合约创建者、合约所有者或由创建者授权过的账户**调用
- 铸币 **mint** 函数根据输入的铸币金额 **wad**，在消息发送方 **msg.sender** 的账户余额以及代币总供应量 **_supply** 的数额上直接增加 **wad** 数额的代币
- 销毁 **burn** 函数则是在账户余额以及代币总供应量中减少相应数额的代币

ERC721 – 标准定义

- ERC 721 合约标准规定了一种**不可替代**的代币（Non-fungible Token, NFT）的合约接口
- 此类代币的**最小单位**为个，即在 ERC 20 标准中对应小数点位的 decimal 值为零
- 此类代币最重要的特点为每一个代币都是**独一无二**的，每一个代币拥有各自的 tokenId 标号，并且可以附上一些各不相同的特征值，这样使得每个代币都是“**不可替代**”的

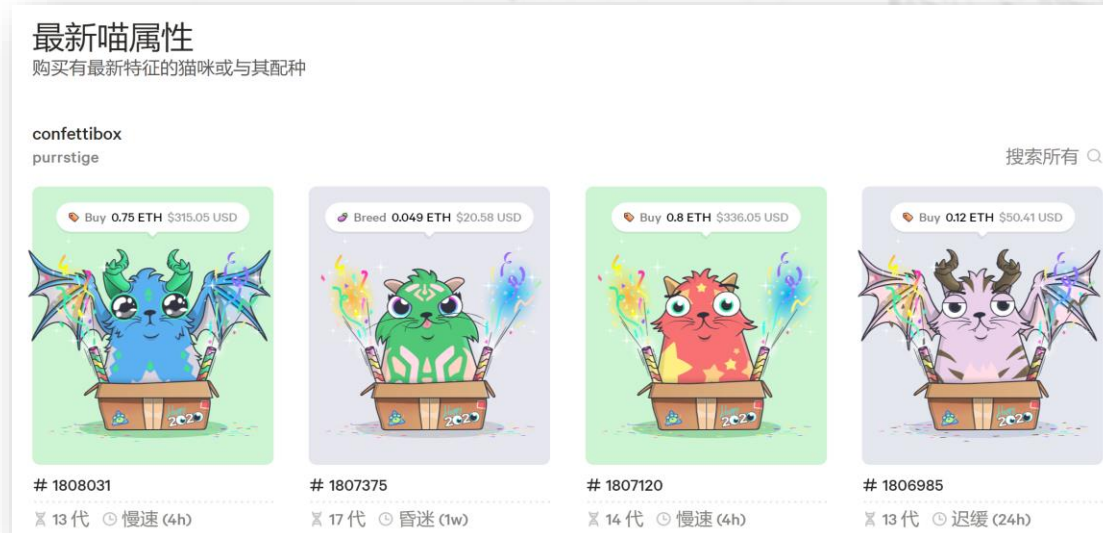
ERC721 –标准接口

```
contract ERC721 {  
    // Required method  
    function totalSupply() constant returns (uint256 totalSupply);  
    function balanceOf(address owner) constant returns (uint256 balance);  
    function ownerOf(uint256 tokenId) constant returns (address owner);  
    function approve(address _to, uint256 _tokenId); function takeOwnership(uint256 tokenId);  
    function transfer(address to, uint256 tokenId);  
    // Optional method  
    function name() constant returns (string name);  
    function symbol() constant returns (string symbol);  
    function tokenOfOwnerByindex(address owner, uint256 index) constant returns (uint tokenId);  
    function tokenMetadata(uint256 tokenId) constant returns (string infoUrl);  
    // Events  
    event Transfer(address indexed _from, address indexed _to, uint256 _tokenId);  
    event Approval(address indexed owner, address indexed _approved, uint256 tokenId);  
}
```


ERC721 – 标准接口

- ERC 721标准**继承**了ERC 20标准的一些基本功能接口，并在一些函数中加入了**`_tokenId`**用以指定特定的代币
- ERC 721 相比于ERC 20 还**新增**了一些功能函数：
 - **`ownerOf`**和 **`tokenOfOwnerByIndex`**分别为根据代币ID 查询该代币的持有者，以及根据持有者及其索引查询所持有的代币ID
 - ERC20中的 **`transferFrom`**方法被更改为**`takeOwnership`**，在限额 **`approve`** 允许的条件下，交易发起方调用该函数可以将指定 **`_tokenId`** 的代币从他人处转至自己的账户中
 - **`tokenMetadata`**函数用于查看代币的元数据等，根据代币的 ID 查询到一个 **URL** 格式字符串，其中包含这一代币的名称、图像和描述等相关信息

ERC721 – 以CryptoKitties为例



- 在游戏中，用户还可以让小猫进行繁殖，后代会产生出全新的基因和外表
- 每只小猫形态各异的特点正是由 **ERC 721** 合约标准中的“不可替代的代币” **NFT** 所实现

ERC721 – 以CryptoKitties为例

- CryptoKitties 合约应用了 ERC 721 标准定义了小猫代币，每个小猫代币拥有独一无二的_tokenId，并且包含基因 genes、出生时间 birthTime、父亲 matronId、母亲 sireId 等信息。
- 合约根据 ERC 721 标准实现了 transfer、approve、ownerOf和 tokenOfOwner等函数功能
- 该合约中仍保留了加上_tokenId 后的 transferFrom函数，相比 takeOwnership函数还能够实现（如向第三方转账等）更多的功能

ERC721 – 以CryptoKitties为例

