**以太坊读书报告**

# 以太坊（Ethereum）

以太坊是一个去中心化平台，可以在其上运行使用智能合约编写的DApp（DApp可以由一个或多个智能合约构建。以太坊智能合约是在以太坊上运行的程序，可以使用Solidity语言编写。智能合约完全按照程序运行，杜绝了停机、中心化操控、欺诈和第三方干涉的可能性。）

使用以太坊运行智能合约的主要优点是方便智能合约彼此交互，而且不需要担心整合共识协议等事情，只需编写应用所需逻辑即可。

以太坊内置了一种加密货币——以太币（ether）。部署智能合约或者执行智能合约函数需要用到以太币。

和任何其他DApp一样，智能合约可以有多个实例，且每个实例都有自己专门的地址。用户账户和智能合约都可以持有以太币。

以太坊使用区块链数据结构和工作量证明共识协议。智能合约可以通过发送交易调用或者其他合约调用。

# 以太坊关键概念

### 账户（Account）

外部账户（Externally Owned Accounts，EOA）

一般意义上的用户账户，用户在创建账户时自动生成公私钥对，编码存放在Keyfile中，私钥使用用户口令加密，公钥哈希值截取后20位作为账户地址

合约账户（Contract Accounts）

保存在以太坊区块链上，是合约代码（功能）和数据（状态）的集合。合约账户通过外部账户或既存合约账户进行部署和操作控制。在部署合约二进制代码时由以太坊虚拟机（EVM）基于创建者账户地址、创建者交易nonce生成账户地址。账户地址生成过程如下图所示。



**交易（Transaction）**

以太坊的交易是外部账户发往其他账户（外部账户或合约账户）报文结构体，只要包括接收者地址（0地址代表新建合约实例）、发送者签名、发送金额、数据域（若接收方为合约账户）、Gas上限和Gas价格。

**消息（Messsage）**

以太坊的消息是合约账户发往其他账户（外部账户或合约账户）报文结构体，主要包括消息发送者、消息接受者地址、发送金额、数据域（若接收方为合约账户）和Gas上限。

**Gas**

由于以太坊为公有链设计，为了避免恶意程序无代价发动Dos攻击或滥用以太坊网络，参考比特币引入了经济概念，所有在以太坊EVM上的操作和存储消耗都需要Gas。

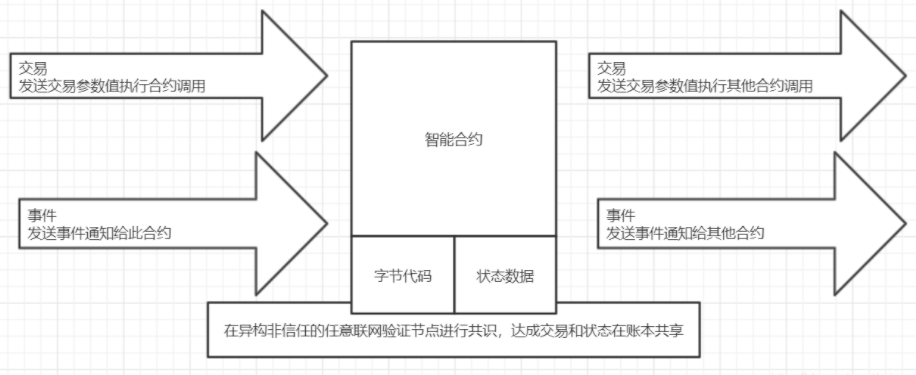
交易发送者设定Gas上限和Gas价格作为矿工打包费用。如果费用过低，矿工可以选择不打包交易到区块。

合约在执行过程中，如果当前代码执行以及子消息产生的代码执行发生的所有Gas消耗超过Gas上限，矿工依然可以获得打包费用，但是当前交易的所有状态都会被回滚。如果未出现Gas耗尽（gas overrun），交易成功执行，所有未消耗的Ether（以太币）依然会返回发送者账户。

矿工挖矿获得Ether，合约执行消耗Ether，形成天然的供求关系。Gas只与程序逻辑处理复杂度有关，将其与市场波动相关的Ether价格隔离开，通过发送者设定Gas Price的方式，可以对抗通货紧缩/膨胀。

**合约（contract）**

以太坊的合约即为合约账户，是以EVM字节码形式存储的代码（功能）和数据（状态）的集合体。它们以Patricia树组织，存储在区块链的账户地址上，账户可以相互发送交易或消息事件，在Gas上限内进行一次交易的图灵完备运算。交易发送者支付实际消耗Gas \* 指定GasPrice的Ether给矿工作为费用，将此交易和结果打包到下一个区块上。智能合约的交互如下图所示。



**以太坊虚拟机（EVM）**

以太坊虚拟机是智能合约的运行环境，是比JVM或Docker VM“沙箱”模式还严格的一种完全隔离的虚拟运行环境。运行在EVM中的合约代码无法访问网络、文件系统和其他进程，合约之间的访问也受到严格限制。EVM是基于栈操作的虚拟机，栈最大深度1024，每个元素32个字节，其指令集由最基本的算术、位、逻辑、比较、条件跳转、无条件跳转这些满足图灵完备的最小指令集组成，EVM可以发送消息调用其他合约，还可以加载目标地址的合约代码在当前合约上下文执行（callocode）

**DApp（去中心化应用）**

DApp直接关联起用户和服务提供方，即买卖双方的应用。以太坊提供多种语言版本的API客户端，并通过以太坊客户端接入网络，DApp基于这些API客户端，把需要共识的业务逻辑和数据以合约的形式部署到以太坊区块链上，并按照需要执行API调用操作。那些不需要共识的逻辑在DApp自身中实现，以降低运行成本。

**以太坊架构**

作为分布式价值交换网络，以太坊要建模的是一个公有的、可以被世界上任何一个联网的人使用的状态机。这个状态机由交易驱动，在其上可以执行图灵完备的应用，可以自动在分布式不信任的互联网上达成共识。

以太坊遵从如下架构原则：

1.三明治复杂度模型

为确保以太坊网络底层尽量简单、以太坊的上层应用接口也尽量简单，将复杂度集中在中间层，如核心共识、高级语言编译器、参数序列化和反序列化脚本、存储数据结构模型、LevelDB存储接口和通信协议等。

2.对应用自由无偏见

不对用户发布的合约和发起的交易有任何限制或鼓励，本着“网络中立”的原则，以太坊“交易手续费”让过度使用网络付出相应的代价。

3.协议和操作码通用性

所有协议特性和EVM操作码都包含尽量少的语义含义，这样可以充分组合使用，满足未来更高级的语义用途。

4.没有提供专门的特性

通用性设计的必然结果就是平台的协议不提供任何特性。用户可以把这些特性通过智能合约的方式实现，如特定的合约代币（token）、侧链。

5.有风险偏好

对于存在引起风险的技术改变，如果可以带来很大的收益，如极大地提高出块速度、提高共识效率，都是可以接受的。

# 以太坊数据模型

**RLP编码**

全称：Recursive Length Prefix（递归长度前缀）

该算法可用来序列化任意嵌套的二进制数组，是以太坊主要的序列化编码算法。

**Merkle Patricia树**

改进版Merkle Patricia树是以太坊主要的数据结构，用来存储键值对，可用于账户状态、交易列表、收据列表等类似字典（dictionary）的数据结构，并进行树形结构的数据表示和存储，对于插入、检索、修改、删除可以实现O ( l o g ( n ) ) O(log(n))O(log(n))搜索复杂度。

**账户模型**

区别于比特币的UTXO模型，以太坊对的数据模型更像用传统的数据库方式存储状态，状态包括账户余额和智能合约账户的各种状态，通过层层组装，最终将以太坊所有账户和对应的状态（包括合约存储子状态哈希）用Patricia树的方式组织成账户模型。

以太坊的所有账户（EOA和合约账户）都需要保存余额。另外，对于合约账户，还需要保存合约账户的各种状态值，把合约账户中所有要记录的状态和对应的值以Patricia树方式构建得到合约账户的状态存储根哈希值（storageRoot），对于外部用户账户EOA，这个值就是空字串。

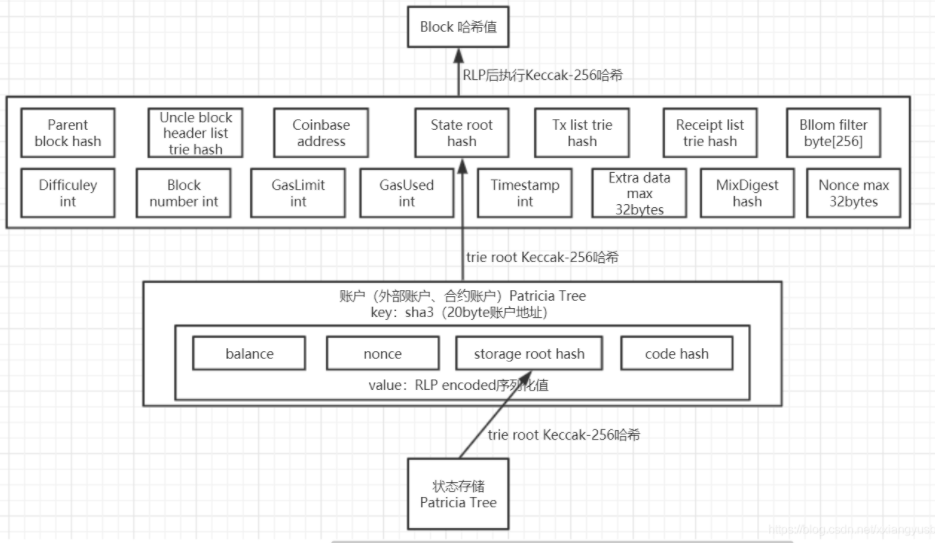
以太坊会将所有账户的余额（balance），账户交易序列号（nonce），账户的状态存储根哈希值（storageRoot），合约代码哈希值（codeHash）组成的元组[nonce, balance, storageRoot, codeHash]进行RLP序列化编码的字符串作为value，以对应账户的20个字符账户地址作为key，再以Patricia树进行组织，计算出世界状态根哈希值，作为当前区块头哈希值的计算要素之一。

**区块结构**

一个区块由区块体[uncleHeaderList, transactionList]和区块头blockHeader构成。

交易列表（Transaction List）是当前打包进区块的所有交易。一个交易可以是外部账户的转账交易，也可以是合约部署交易，还可以是合约函数的执行交易。

区块头结构和生成过程如下图所示：



# 参考资料

[1] 邹均,于斌,庄鹏,邢春晓.区块链核心技术与应用[M].北京:机械工业出版社,2018

[2] Narayan Prusty.区块链项目开发指南[M].北京:机械工业出版社,2018

[3] 袁勇,王飞跃.区块链技术发展现状与展望[J].自动化学报,2016,42(04):481-494

[4] Wood, Daniel Davis. “ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER.” (2014).