SET是什么币?SET币全称Pointset，SET币中文名称点集网络。点集(PointSet)作为一个底层网络，以分布式技术为基础，构建一个去中心化网络体系。从底层数据多渠道获取和高性能存储，到中间层数据的处理和链上记录，再到应用层提供 api 和 sdk，为开发者和用户构建底层数据与上层应用的桥梁，在信息交换和价值转移上发挥着重要的作用。通证经济模型下的 SET
作为整个生态的价值转移媒介，多层挖矿机制能够保证整个生态的良性循环。

发行时间：2018-04-03

最大供应量：21,000,000,000 SET

总供应量：21,000,000,000 SET

上架交易所：1家

VVBTC交易所：https://www.vvbtc.com/

PointSet 链应用最新一代技术开发，为普通用户、开发者、网站、第三方伙伴、平台、组织等多元渠道提供面面俱到的链上数据记录(身份认证、版权保护、激励机制等)。 PointSet 链上每个数据都是一个 Point，无数个 Point 构成了一个
Set。链上包含智能合约，开发者能快速搭建自己的
DAPP;也可以通过点集网络上层的 PointDock
快速集成。除了现有的主链技术外，PointSet 还自创改进了包括 PointCheck、PointDock、PointPaxos 在内的一系列基础设施，为生态数据的安全保驾护航。

区块 SET 块的构成大量借鉴了优秀主链的构造，包含以下部分内容：

a) ParentHash 父区块的哈希

b) stateRoot：当前已定稿区块的交易组成的状态数根节点的哈希

c) transactionRoot: 交易树根节点的哈希

d) receiptRoot：收据树根节点的哈希

e) logsBoom：所有交易收据中的可索引信息组成的布隆过滤器

f) difficulty：打包当前区块的难度纯量值

g) number：区块的祖先的数量

h) timestamp：区块初始化的时间戳

i) extraData：对当前区块的备注

j) mixHash：256 位哈希

k) nonce：64 位值(和 mixHash 共同证明计算量的承载是否足够)

其中交易树和收据树都是 Merkle 树，默克尔树：

Merkle Tree 可以看做 Hash List 的泛化(Hash List 可以看作一种特殊的 Merkle Tree，即树高为 2 的多叉 Merkle Tree)。

在最底层，和哈希列表一样，我们把数据分成小的数据块，有相应地哈希和它对应。但是往上走，并不是直接去运算根哈希，而是把相邻的两个哈希合并成一个字符串，然后运算这个字符串的哈希，这样每两个哈希就结婚生子，得到了一个”子哈希“。如果最底层的哈希总数是单数，那到最后必然出现一个单身哈希，这种情况就直接对它进行哈希运算，所以也能得到它的子哈希。于是往上推，依然是一样的方式，可以得到数目更少的新一级哈希，最终必然形成一棵倒挂的树，到了树根的这个位置，这一代就剩下一个根哈希了，我们把它叫做 Merkle Root。

在 p2p 网络下载网络之前，先从可信的源获得文件的 Merkle Tree 树根。一旦获得了树根，就可以从其他从不可信的源获取 Merkle tree。通过可信的树根来检查接受到的 Merkle Tree。如果 Merkle Tree 是损坏的或者虚假的，就从其他源获得另一个 Merkle Tree，直到获得一个与可信树根匹配的 Merkle Tree。

Merkle Tree 和 Hash List 的主要区别是，可以直接下载并立即验证 Merkle Tree 的一个分支。因为可以将文件切分成小的数据块，这样如果有一块数据损坏，仅仅重新下载这个数据块就行了。如果文件非常大，那么 Merkle tree 和 Hash list 都很到，但是 Merkle tree 可以一次下载一个分支，然后立即验证这个分支，如果分支验证通过，就可以下载数据了。而 Hash list 只有下载整个 hash list 才能验证。

默克尔树能快速的定位树叶的改变，大量节省查询耗时。而交易状态树是默克尔帕特里夏树(MPT)：

MPT(Merkle Patricia Tree)顾名思义，MPT
就是默克尔树和葩特里夏树的混合体：

在 SET 链中，我们使用一种十六进制的前缀编码，字母表中存在 16
个字符，其中已一个字符为 nibble MPT
树中的节点包括空节点、叶子节点、扩展节点和分支节点:

空节点，简单的表示空，在代码中是一个空串。

叶子节点(leaf)，表示为[key,value]的一个键值对，其中 key 是 key
的一种特殊十六进制编码，value 是 value 的 RLP 编码。

扩展节点(extension)，也是[key，value]的一个键值对，但是这里的 value
是其他节点的 hash 值，这个 hash
可以被用来查询数据库中的节点。也就是说通过 hash 链接到其他节点。

分支节点(branch)，因为 MPT 树中的 key 被编码成一种特殊的 16
进制的表示，再加上最后的 value，所以分支节点是一个长度为 17 的 list，前 16
个元素对应着 key 中的 16 个可能的十六进制字符，如果有一个[key,value]对在这
个分支节点终止，最后一个元素代表一个值，即分支节点既可以搜索路径的终止也
可以是路径的中间节点。

MPT 树中另外一个重要的概念是一个特殊的十六进制前缀(hexprefix,
HP)编码，用来对 key 进行编码。因为字母表是 16 进制的，所以每个节点可能有
16 个孩子。因为有两种[key,value]节点(叶节点和扩展节点)，引进一种特殊的终止
符标识来标识 key 所对应的是值是真实的值，还是其他节点的
hash。如果终止符标记被打开，那么 key 对应的是叶节点，对应的值是真实的
value。如果终止符标记被关闭，那么值就是用于在数据块中查询对应的节点的
hash。无论 key 奇数长度还是偶数长度，HP
都可以对其进行编码。最后我们注意到一个单独的 hex 字符或者 4bit
二进制数字，即一个 nibble。

HP 编码很简单：一个 nibble 被加到 key 前(下图中的 prefix)，对终止符的状态和
奇偶性进行编码。最低位表示奇偶性，第二低位编码终止符状态。如果 key
是偶数长度，那么加上另外一个 nibble，值为 0 来保持整体的偶特性。