

实验一：区块链浏览器-数据的获取与解析

一 实验概述

区块链技术的重要特点之一是具有数据不可篡改性。而在其所构筑的公链应用（如数字货币、智能合约）中，良好的数据透明性使得经过区块链接收确认的所有数据变得公开可验证，这也是区块链技术无需信任的决定性因素。区块链浏览器，作为区块链项目的关键基础设施，能够帮助大众在无需运行任何专用软件的情况下，对实时的区块链状态进行解析，获取其感兴趣的部分数据，也是学习区块链技术最为直观便捷的工具。

本实验以比特币和以太坊的区块链浏览器为例，首先介绍获取区块链数据的基本技巧；进而利用区块链浏览器解析并学习区块链账本层与合约层的构造，结合多个典型事务，加深学生对于多种区块链状态的认知与体会；最后对批量获取区块链数据进行数据挖掘的相关技巧进行点拨。该实验也是后续所有区块链实践的必备积淀。

实验内容概述如下：

- A. 掌握区块链浏览器的基本操作、功能与使用技巧（各类状态查询，简单API调用，数据可视化，钱包，测试链）
- B. 利用区块链浏览器解析并学习区块链账本层构造（地址、典型交易、交易费用、隔离见证、脚本构造等）
- C. 利用区块链浏览器解析并学习区块链合约层构造（合约状态、合约的相互调用、费用计算、ERC20等）
- D. 拓展实验：批量获取并分析区块链元数据（API调用/爬虫的使用、数据挖掘/部署开源区块链浏览器）

二 预备知识

1. **区块链浏览器**：作为浏览区块链信息的主要窗口，区块链浏览器向用户提供其感兴趣的区块链相关信息，包括但不限于：链状态、区块状态、交易状态、合约状态、账户状态，并且区块链浏览器还可能额外提供对于测试网络的支持（方便开发者进行测试应用的调试），数据可视化服务（方便用户对区块链状态进行宏观认识），钱包服务（方便用户管理数字资产）和开放 API（方便用户精确、批量的获取数据）。

TIPS：一些主流且稳定的区块链浏览器：

Blockstream: <https://blockstream.info/>

Blockchain: <https://www.blockchain.com/explorer>

BTC: <https://btc.com/block>

Blockcypher: <https://live.blockcypher.com/>

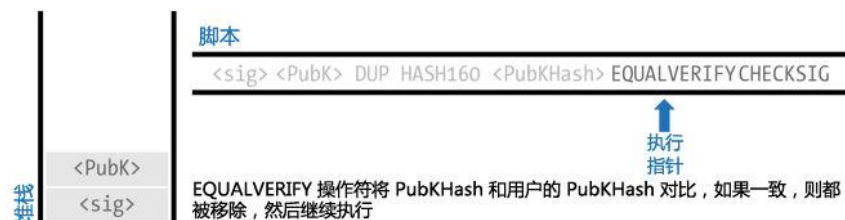
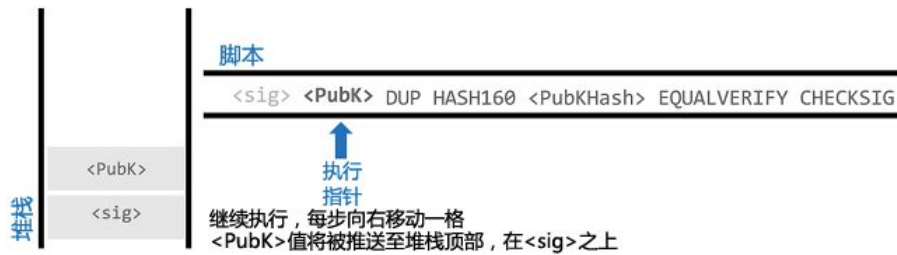
Etherchain: <https://www.etherchain.org/>

Etherscan: <https://etherscan.io/>

2. **Base58**：用于 Bitcoin 中使用的一种独特的编码方式，主要用于产生 Bitcoin 的账户地址。相比于 Base64 去除了易于混淆的 00II 和作为标点的+/,使得地址的表述更加清晰，但该种编码方式也因此付出了编码效率上的妥协。

3. **比特币脚本**：是比特币使用的一种基于堆栈的执行语言，比特币将一系列带有特定功能的执行脚本 OPCODE 进行特定编码，通过合理组合后按照“先入后出”的顺序执行，辅助完成交易的验证功能。

我们以最常见的 P2PKH 为例，演示一下以其为核心脚本的账本交易验证过程-因为 P2PKH 是支付给收款者公钥的 hash, 所以交易的验证脚本需要分两步完成：首先验证交易发布者所提供的公钥在哈希运算后是否匹配；其次通过该公钥验证交易发布者提供的签名，若两者皆成功，则该输入的花销是有效的，如图所示：



所有的比特币脚本都可以在 <https://en.bitcoin.it/wiki/Script> 中进行查询。

4. **API: Application Programming Interface**, 即应用程序接口, 规定了运行在一个端系统上的软件请求因特网基础设施向运行在另一个端系统上的特定目的地软件交付数据的方式。API 的使用使得软件系统的职责得到合理划分, 有助于提高系统的可维护性和可扩展性。

5. **JSON: JavaScript Object Notation**, 是一种常用的轻量级数据交换格式, 因其具有简洁和清晰的层次结构, 而成为理想的数据交换语言, 既易于人阅读和编写, 同时也易于机器解析和生成, 并有效地提升网络传输效率。

如以下示例, 其语法为: 1) 在 JS 语言中, 一切都是对象;

2) 以花括号保存对象, 以键值对表示对象, 以方括号保存数组, 数据由逗号分隔;

3) 键/值对组合中的键名写在前面并用双引号 "" 包裹, 使用冒号 : 分隔, 然后紧接着值。

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
```

```

    "number": "646 555-4567"
  },
  {
    "type": "mobile",
    "number": "123 456-7890"
  }
],
"children": [],
"spouse": null
}

```

6. 智能合约与账本交易的基础知识请参考同期的区块链课程教义。

三 实验准备

实验系统：任意；

使用软件：最新版本 Chrome 浏览器 ；

其他：鉴于访问速度和实验体验，有条件的同学可以选择开启代理。

四 实验内容

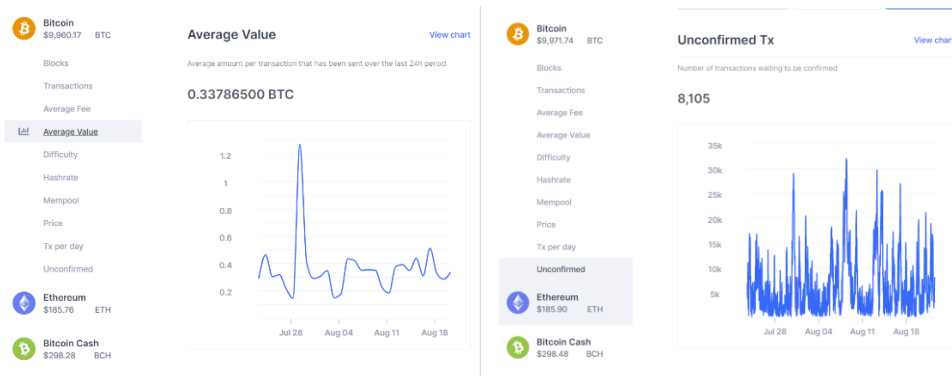
3.1 区块链浏览器的基本操作与功能

本节以几种常用区块链浏览器为例，介绍区块链浏览器的基本功能：

a. 首先介绍浏览器反馈的几类区块链状态：

1) 链状态：每一条区块链的状态由其链名称作为唯一标识；

打开 <https://www.blockchain.com/explorer?currency=BTC&stat=blocks>

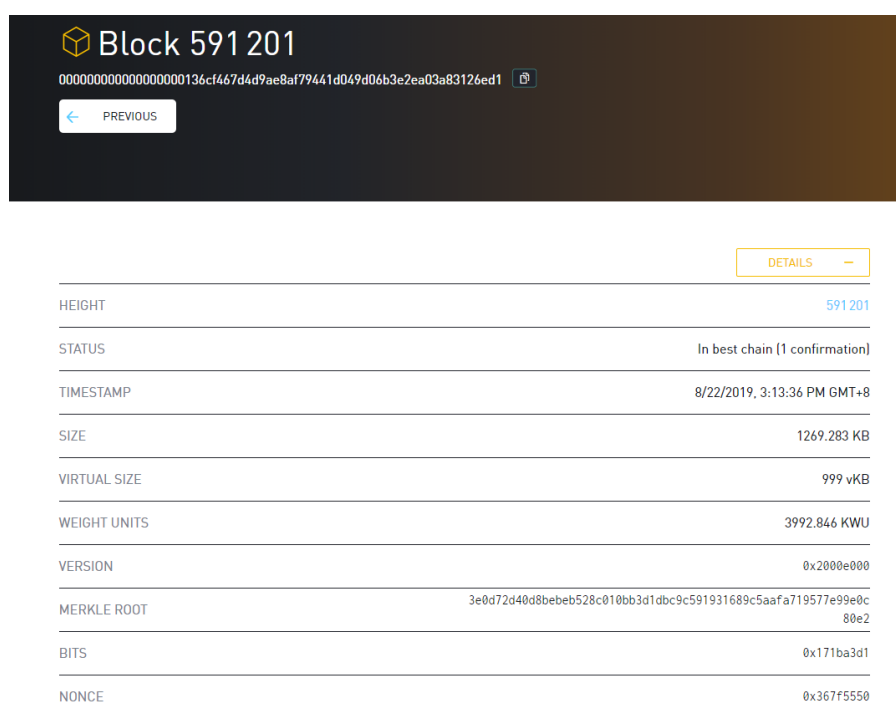


可以访问 Bitcoin, Ethereum, Bitcoin Cash 三条链的状态；其主要特征包括：

链的最新区块、最新确认交易池、平均交易费用、平均交易价值、实时挖矿难度、全网节点总算力、待确认交易池、代币价值、每日交易频率以及积压的交易总数目，由此可以初步判断该条链的价值、效率、安全性以及交易的活跃度。

2) 区块状态：以区块地址和区块高度作为标识；



打开 <https://blockstream.info>，点击任意一个区块地址，以 <https://blockstream.info/block/00000000000000000136cf467d4d9ae8af79441d049d06b3e2ea03a83126ed1?expand> 为例：



Block 591201	
00000000000000000136cf467d4d9ae8af79441d049d06b3e2ea03a83126ed1	
PREVIOUS	
DETAILS	
HEIGHT	591201
STATUS	In best chain [1 confirmation]
TIMESTAMP	8/22/2019, 3:13:36 PM GMT+8
SIZE	1269.283 KB
VIRTUAL SIZE	999 vKB
WEIGHT UNITS	3992.846 KWU
VERSION	0x2000e000
MERKLE ROOT	3e0d72d40d8bebeb528c010bb3d1dbc9c591931689c5aafa719577e99e0c80e2
BITS	0x171ba3d1
NONCE	0x367f5550

其主要特征包括：状态（有无分叉，确认深度），时间戳（并非一个精确的值，仅仅具有参考意义），实际数据大小，可见大小（一般为 Bitcoin 规定的 1MB，节约的数据量由账本层隔离见证机制的施行带来，我们在下一节着重解析），由隔离见证带来的额外数据（以 KWU 为单位，采用独特的换算标准 https://en.bitcoin.it/wiki/Weight_units），矿工节点版本号，区块的默克尔根，以及生成有效 PoW 所需要有效填充数 Nonce。区块详情中则包含了该区块容纳的所有交易信息，其中第一个交易固定为 Coinbase 类型，作用是将挖矿奖励支付

3) 交易状态：以交易地址作为唯一标识，点击任何一个交易，以 <https://blockstream.info/tx/e75cc9e67a64d3974210da8480d3d80c0b5fb1a966b6451dd847754d4e82a5e1> 为例，其主要特征包括：确认状态、所在的区块信息（地址，高度，时戳）、所付出的交易费用、交易的大小、节点版本号、锁定时间（用于定义该交易最早可入块的时间）、费用节约情况（如果激活隔离见证能够节省的费用）以及隐私情况（是否重用地址等）。进一步，通过点击交易的详情，我们可以观察交易的构造以及其每个输入的赎回脚本 **ScriptSig**，每个输出的锁定脚本 **ScriptPubkey** 和输出的花费情况。

<div>  Transaction </div> <div> e75c9e67a64d397421dda8480d3d80c0b5fb1a966b451dd847754d4e82a5e1  </div>	
STATUS	7 Confirmations
INCLUDED IN BLOCK	0000000000000000000129be115b4d45ef5d8ff2a662db69d51 02998b8f27d7
BLOCK HEIGHT	59122
BLOCK TIMESTAMP	8/22/2019, 6:56:42 PM GMT+0
TRANSACTION FEES	0.001 BTC (283.3 sat/vB)
SIZE	353 B
VIRTUAL SIZE	353 vB
WEIGHT UNITS	1412 WU
VERSION	2
LOCK TIME	0
SEGWIT FEE SAVINGS	This transaction could save 29% on fees by upgrading to SegWit native SegWit-Bech32 or 23% by upgrading to SegWit P2SH
PRIVACY ANALYSIS	Address reuse


4) 账户状态：每个用户账户由其地址唯一标识，点击任意一个输入或输出中包含的地址字段，以

<https://blockstream.info/address/3NKtXY8ZpZe5XbE4YrjZogkid8hSBkDACw>

为例，可以检索该地址相关的所有交易历史、以及地址的余额。

TIPS: 对于比特币的账本交易，不建议通过地址重用的形式来管理账户，因为这样容易因为交易历史而暴露隐私，另外由于地址上存储的资产在花销后，其对应的公钥也就随之泄露，一旦进入后量子密码时代，旧公钥密码算法的失效将会直接导致用户的资产流失。目前最为安全的账户管理方法是：钱包记录并衍生一系列的用户地址，每个地址仅仅使用一次，钱包采用过滤器监听所有相关地址的交易并整理用户资产，避免上述问题的出现。

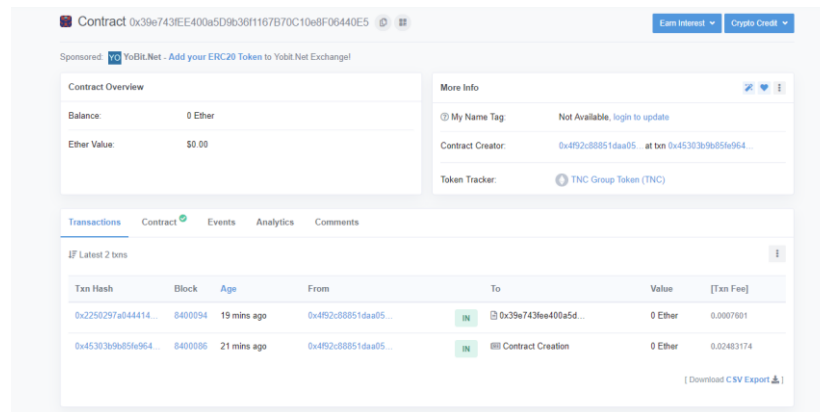
Address

3NKtXY8ZpZe5XbE4YrjZogkid8hSBkDACw 

CONFIRMED TX COUNT	3
CONFIRMED RECEIVED	2 outputs (0.03830701 BTC)
CONFIRMED SPENT	1 output (0.022167 BTC)
CONFIRMED UNSPENT	1 output (0.01614001 BTC)

5) 合约状态：以合约地址作为唯一标识，访问以太坊的区块链浏览器 Etherscan ， 选 择 其 中 的 任 意 有 效 合 约 进 行 观 察 ：
<https://etherscan.io/contractsVerified>，以

<https://etherscan.io/address/0x39e743fee400a5d9b36f1167b70c10e8f06440e5> 为例，观察中可以辨识的特征有：合约名称，该合约的所有相关事务（初始的两笔事务分别用于创建合约以及向合约地址付款以启动合约），源码，账户余额，创建者地址，编译版本，遵循的协议，状态变更历史以及合约提供的接口（ABI）等众多信息。



以上状态皆可通过区块链浏览器的搜索栏，输入相应的标识进行查询，这也是浏览器提供的最为基本的功能。

b. 其次介绍区块链浏览器提供的 API 支持

以 blockstream 为例，其 API 调用的完整方法记录在：

<https://github.com/Blockstream/esplora/blob/master/API.md#transaction-format>

中；使用 API 的方法有两类，第一类是使用 js 包管理工具直接对该开源浏览器进行部署；另外一种较为简单的方法则是直接通过 url 访问 API（前缀为 <https://blockstream.info/api/>），我们建议使用命令行工具 curl，一款利用 URL 语法在命令行方式下工作的开源文件传输工具完成调用（采用浏览器直接访问亦可）：

我们以其中的第一条 GET /tx/:txid/status 为例，尝试调用该接口：

该接口的功能是根据交易的地址，返回交易的确认状态，包括：是否被确认 (confirmed)，被收纳的区块高度 (block_height) 以及该区块的地址 (block_hash)。

例如查询地址为

6dcc37358d08b6adee18deb22f037326b5e659c2030189fbf774344c9fb3915 的交

易确认状态，打开终端，输入：

```
curl
https://blockstream.info/api/tx/6dcc37358d08b6adee18deb22f037326b5e659c203018
9fbf774344c9fb3915/status
```

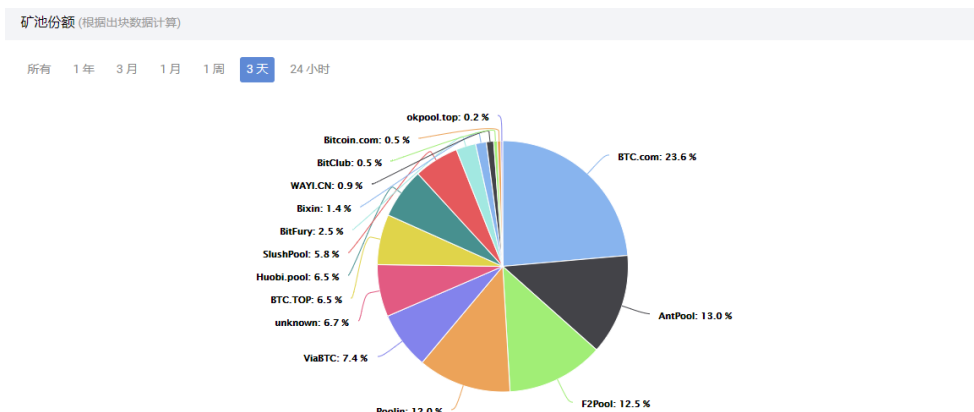
```
C:\Users\vivid>curl https://blockstream.info/api/tx/6dcc37358d08b6adee18deb22f037326b5e659c2030189fbf774344c9fb39152/status
{"confirmed":true,"block_height":364292,"block_hash":"000000000000000003dd2fdbb484d6d9c349d644d8bbb3cbfa5e67f639a465fe",
"block_time":1436293147}
C:\Users\vivid>
```

如上图可见，所查询交易的确认状态以 json 的形式被返回，安装 python 的同学可以使用 `python -m json.tool` 和管道对返回数据的格式进行修整。

```
C:\Users\vivid>curl https://blockstream.info/api/tx/6dcc37358d08b6adee18deb22f037326b5e659c2030189fbf774344c9fb39152/status|python -m json.tool
100 144 100 144 0 0 144 0 0:00:01 --:--: 0:00:01 263
{
  "confirmed": true,
  "block_height": 364292,
  "block_hash": "000000000000000003dd2fdbb484d6d9c349d644d8bbb3cbfa5e67f639a465fe",
  "block_time": 1436293147
}
```

c. 一些浏览器对区块链的历史数据进行了进一步的挖掘，并提供了可视化的服务，方便用户对链状态的变化进行探究，我们以 btc 浏览器对比特币提供的多项统计为例：

打开 <https://btc.com/stats>，可以看到其对于挖矿份额、交易费用、脚本类型、难度变更等多项参数进行实时的统计，以矿池份额为例，如下图：我们都知道，Nakamoto Consensus 理论上能容忍的恶意节点比例为 51%，在考虑自私挖矿后，这一值被降低至 25%，而现有最大矿池的算力占比已经接近这个值，矿池的扩张正在逐渐蚕食比特币的安全性。

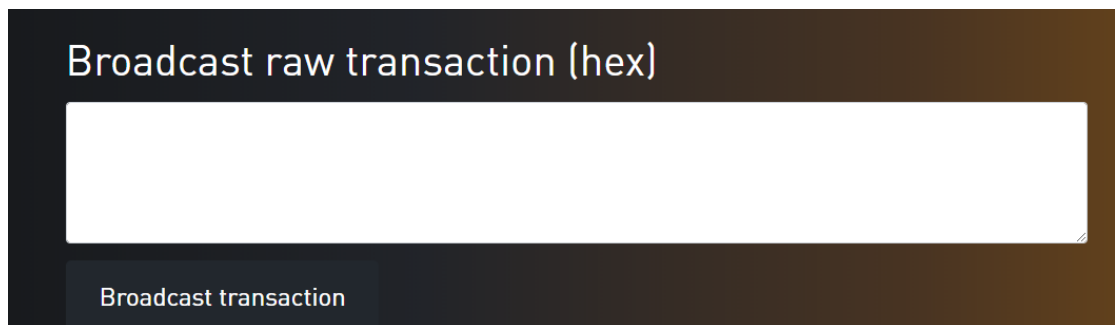


另一方面，去中心性的丧失也是令人担忧的，图中非矿池的算力占比仅占 6.7%，而对比整个比特币挖矿历史，这一占比为 37.4%，庞大的算力消耗与存储

需求已使得个人节点不再适合作为全节点维护比特币生态,更倾向于加入矿池承担 hash puzzle 的运算外包业务。

d.一些区块链浏览器提供了网页端发布交易,甚至是数字货币钱包的功能:

例如在 <https://blockstream.info/tx/push>: 输入交易的 hex 编码,浏览器就可以代理广播该交易;



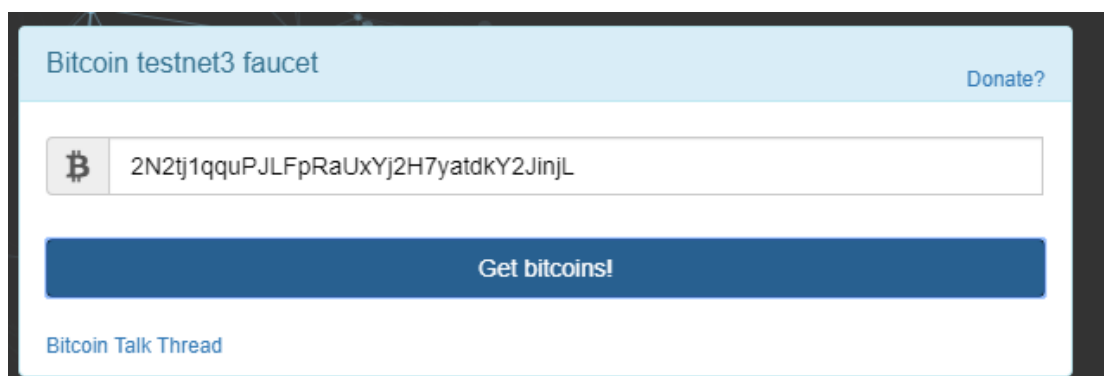
在 <https://wallet.btc.com>, <https://www.blockchain.com/wallet> 皆提供了钱包注册的功能。

e.测试网络支持

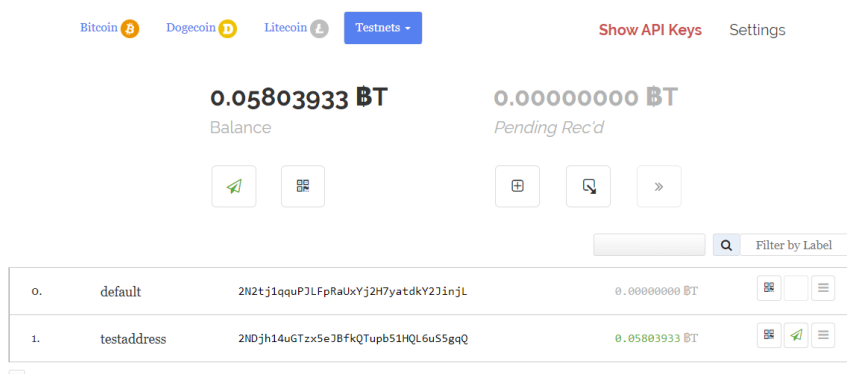
各个区块链社区考虑到了代币的昂贵,建立了测试生态用于开发者对于区块链应用的测试,如比特币的 Testnet3,其特点是挖矿难度很低,代币没有经济价值,但除此之外,其部署的技术一般优先于比特币主网。

blockstream 和 blockcypher 都提供了对于测试网络的浏览器支持,我们以 blockstream 为例: <https://blockstream.info/testnet/>。

测试网络的代币可以从各类 testnet faucet 上输入自己的测试链地址获取:例如 <https://coinfaucet.eu/en/btc-testnet/> (强烈建议在实践结束后将该测试币发送回原 faucet 地址,便于更多人进行实验!)



更可贵的是，block.io 为测试网络提供了钱包的支持，我们可以在其上进行注册，并使用测试网络代币管理账户，发送交易，体验丰富的钱包功能，定制并观察自己的交易。（完成此实践需要等待较长的确认时间，建议有兴趣的同学在课下实践）



练习 1: 1) 请在区块链浏览器中查询区块

00000000000000000003dd2fdbb484d6d9c349d644d8bbb3cbfa5e67f639a465fe 并

对该区块进行分析，该区块有何异常，造成该异常的原因是什么，这可能体现了区块链系统设计中的哪些问题？

2) 观察浏览器对于比特币挖矿难度变化的可视化实时结果

<https://btc.com/stats/diff>，尝试回答：难度调整的间隔；难度变化的趋势和其带来的影响；推测平均算力的计算方法。

3) 参考 blockstream API 的调用说明：

<https://github.com/Blockstream/esplora/blob/master/API.md>，调用 API 回答以

下问题：

- a. 当前比特币待验证的交易数目为多少？数据量为多大？大概几个区块才能处理完这些交易？
- b. 给出当前交易费率相对于预计确定区块数目的估计；
- c. 给出高度在 9991-10000 间区块内包含的总交易数目。

3.2 利用区块链浏览器学习区块链账本层构造

本节我们利用区块链浏览器，观察一些典型的账本交易构造方法,学习隔离见证交易的构造方法，进而实践比特币脚本的书写。

首先来熟悉一些典型的交易构造：

- a. Coinbase: 示例- 创始块

<https://blockstream.info/block/00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048>

作为一种特殊的交易固定作为每个区块的第一个交易，将挖矿奖励发送到矿工指定的地址或脚本，其输入脚本无需包含任何典型的赎回脚本，亦没有固定的格式，矿工通常使用的 Pushbytes 可能是嵌入具有一定含义的信息，也有可能是在规定 Nonce 尝试挖矿无果溢出后利用此种办法变相扩展 Nonce 的范围。

#0 Coinbase	
SCRIPTSIG (ASM)	OP_PUSHBYTES_4 ffff001d OP_PUSHBYTES_1 04
SCRIPTSIG (HEX)	04ffff001d0104
NSEQUENCE	0xffffffff

- b. P2PKH: 示例-

<https://blockstream.info/tx/0de586d0c74780605c36c0f51dcd850d1772f41a92c549e3aa36f9e78e905284>

在隔离见证激活前最为常用的一种支付脚本，我们已经在背景中介绍了其验证的过程，锁定脚本格式为： `OP_DUP OP_HASH160 OP_PUSHBYTES_20 <hash>`

OP_EQUALVERIFY OP_CHECKSIG, 其中推送的数据为 20 字节的公钥哈希, 赎回脚本格式为: OP_PUSHBYTES_72 <Sig> OP_PUSHBYTES_33 <Pubkey>, 即先推送 71 或 72 字节的签名, 再推送 33 字节的公钥。

类似的支付脚本有直接向公钥支付的 P2PK, 但由于隐私的原因被 P2PKH 所替代。

c. NullData (OpReturn) : 示例-

<https://blockstream.info/tx/56a3de9926f1d1334b4f76ea9059d8357664d3ab72508b7c35efd9b511d82a01>

作为一种特殊的输出脚本, 可以在交易中嵌入最多 80 字节的任意数据, 该输出不可花销, 亦无法单独作为交易的输出。一般用作保证该部分数据的不可篡改性和时效性, 作为存在性证明进行使用, 辅助搭建更复杂的去中心化应用。

#0 OP_RETURN	0 BTC
TYPE	OP_RETURN
SCRIPTPUBKEY (ASM)	OP_RETURN OP_PUSHBYTES_20 6f6d6e69000000000000001f00000020c25f9f00
SCRIPTPUBKEY (HEX)	6a146f6d6e6900000000000000001f00000020c25f9f00
OP_RETURN DATA	omni ☐ ⬇ ⬆

到这里我们可以回顾账本层交易费用的设定, 其费用与交易的容量呈正比, 这也就意味着在 Bitcoin 这样一个以数字货币功能作为核心的生态内锚定数据是异常昂贵的, 并且会影响其他正常交易的入块, 这也是该脚本设置数据限定的原因。-例如: 示例交易为了锚定 20 字节的数据花费了 0.000446BTC 的费用-当前价值 5 美金。

d.P2SH: 示例-

<https://blockstream.info/tx/d3adb18d5e118bb856fba4b1af936602454b44a98fc6>

[c823aedc858b491fc13?expand](#)

比特币的脚本通过组合可以完成更加复杂的逻辑，构造一些简单的合约，而 P2SH 脚本是实现这类功能最为安全的方法，如下图：P2SH 的锁定脚本构造十分简单，仅包含数据段为赎回脚本的哈希值。

#0 3BnZYLFFeGAAa4aTavSUhez7nAAAnjAJpB	0.0996 BTC
TYPE	P2SH
SCRIPTPUBKEY (ASM)	OP_HASH160 OP_PUSHBYTES_20 6e bdbaf0840274a6b23b0643b75ef4e 8e24f37b8 OP_EQUAL
SCRIPTPUBKEY (HEX)	a9146ebdbaf0840274a6b23b0643b 75ef4e8e24f37b887
SPENDING TX	Spent by e36e89b42a30311fa9d06 24bf5b83c6d5e10a4a180ab5fe42d d1b1f6d99f1891:0 in block #23 2734

但要合法的花费该输出，交易发布者不仅需要在下个交易的输入中嵌入完整的赎回脚本，还需要提供解锁赎回脚本的相应数据，以这个输出为例，最终被 <https://blockstream.info/tx/d3adb18d5e118bb856fbea4b1af936602454b44a98fc6c823aedc858b491fc13?expand> 花费，其赎回脚本：

```
OP_PUSHNUM_2                                OP_PUSHBYTES_65
04f3d35132084eb1b99b6506178c20adb42d26296012e452e392689bdb6553db33ba24b90000
0892805de1646821c7b0fb50b3d879c26e2b493b7041e6215356a0    OP_PUSHBYTES_65
04ab4ecc9e8ea2da0562af25bcaede00c4d5a00db60edc17672376decf0a35a34fdc9f1ffad1fb74
fd7b1b198b9231c25df88e0769bec49975649b4b3f40adafb0        OP_PUSHBYTES_65
04f7149f270717c00f6cc09b9ce3c22791c4aab1af40a5107aacca85b6f644cc0d84459e308f998
d801b8d9d355f8ec33b0e41866841e2870754cf667a9821703d        OP_PUSHNUM_3
OP_CHECKMULTISIG
```

定义了一个 2/3 门限交易，根据脚本 OP_CHECKMULTISIG 的定义，在其脚本中包含了 3 个公钥，而花费该笔输出需要提供对于两个不同公钥的合法签名，OP_CHECKMULTISIG 会利用所有公钥对输入的签名进行验证，满足条件则输出为真。

解锁脚本首先推送解锁赎回脚本所需的数据，然后使用 PUSHBYTES 脚本推送赎回脚本的 HEX 编码。

练习 2：观察 P2SH 交易的赎回脚本：

```
OP_3DUP OP_ADD OP_PUSHDUM_9 OP_EQUALVERIFY OP_ADD OP_PUSHDUM_7  
OP_EQUALVERIFY OP_ADD OP_PUSHDUM_8 OP_EQUALVERIFY OP_PUSHDUM_1
```

参考 <https://en.bitcoin.it/wiki/Script>

说明该脚本规定的解锁条件和运行机理，并拟写其解锁脚本

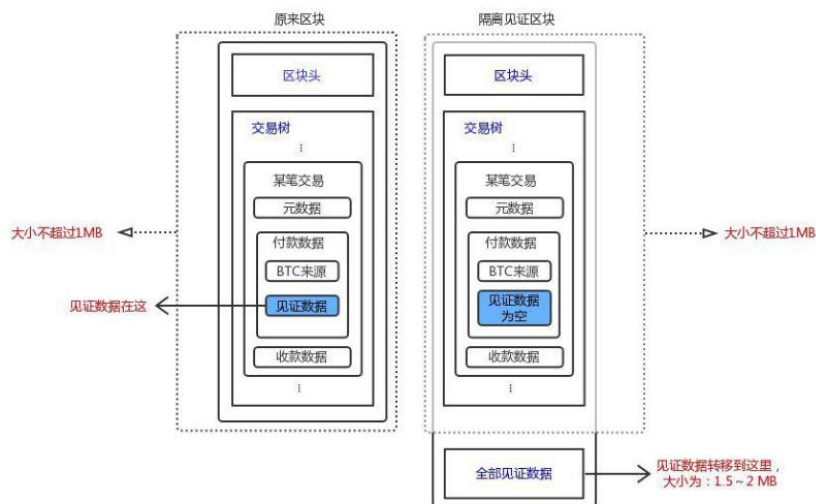
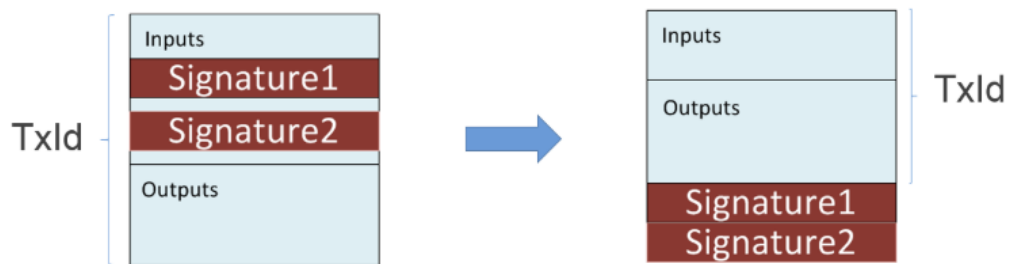
提示：

Word	Opcode	Hex	Input	Output	Description
OP_EQUAL	135	0x87	x1 x2	True / false	Returns 1 if the inputs are exactly equal, 0 otherwise.
OP_EQUALVERIFY	136	0x88	x1 x2	Nothing / fail	Same as OP_EQUAL, but runs OP_VERIFY afterward.
OP_AND	132	0x84	x1 x2	out	Boolean and between each bit in the inputs.
OP_3DUP	111	0x6f	x1 x2 x3	x1 x2 x3 x1 x2 x3	Duplicates the top three stack items.

脚本-hex 转化工具：<https://bc-2.jp/tools/txeditor2.html>

e. 隔离见证

观察之前的账本交易，其占用最多空间的往往是解锁脚本，由于其内部包含的签名以及公钥字段，只在交易验证的一刻起作用，本身不包含其他影响区块链结构或组建交易关系网络的额外信息。因此 15 年，有学者提出隔离见证的思路，如下图所示，将签名与公钥字段从交易的解锁脚本中分离，作为交易的“见证数据”，以独立的数据结构存储。



对于区块而言，由于见证数据不计算在区块容量的限定之中，每个区块可以容纳更多的交易，验证完区块正确性后，见证数据也可以被抛弃，降低对矿工的存储要求。此外，在 BIP0173 中，隔离见证还采用了更加紧致的 Base32 编码代替原来的 Base58 编码，降低了地址编码所占用的空间。

以

<https://blockstream.info/tx/4ef47f6eb681d5d9fa2f7e16336cd629303c635e8da51e425b76088be9c8744c?expand> 为例，我们来观察隔离见证交易的两种经典构造方法：

P2WPKH 和 P2WSH。

该交易的第一个输出为 P2WPKH，其锁定脚本

`OP_0 OP_PUSHBYTES_20 e8df018c7e326cc253faac7e46cdc51e68542c42`

简化了验证所需的操作符，仅仅包含接收者的公钥哈希，OP_0 则代表了隔离见证版本号，以便于直接识别并套用验证模式（类似 P2PKH）。

在花费该输出的交易中，解锁脚本被定义为空，见证数据段则编码了验证所需的签名和公钥。

#5	4ef4716eb681d5d9fa2f7e16336cd629303c635e8d	0.00014293 BTC
	a51e425b76088be9c8744c:0	
WITNESS	304502210088a0b17c0f7db8e2631 df6d48b24a80e1d52f4643e1d3ee2 c3d47b1703e30c5f0220466e1c224 164e6ecac4cc4ae81ff15452cd5dc 8f4fa17fbd9cf842756f4a5b6101 02e31b5276df85870747b1f330e5b 48449fea9903c315c802d040f1ced 036d3a19	
NSEQUENCE	0xffffffff	
PREVIOUS OUTPUT SCRIPT	OP_0 OP_PUSHTOBYTES_20 e8df018c 7e326cc253faac7e46cdc51e68542 c42 (v0_p2wpkh)	
PREVIOUS OUTPUT ADDRESS	bc1qar0srrrr7xfkvy516431ydnw9r e59gtzwwf5mdq	

该交易的第二个输出为 P2WSH，类似于 P2SH，其输出指向赎回脚本的哈希

OP_0	OP_PUSHTOBYTES_32
c7a1f1a4d6b4c1802a59631966a18359de779e8a6a65973735a3ccdfabc407d	

观察花费该输出的交易，可以发现其赎回脚本的功能为 2/2 的门限验证，但对比于之前同样功能的交易方式，由于脚本、签名和公钥数据都被作为见证数据。该交易节约了 52% 的空间，更加的经济且高效。

<https://blockstream.info/tx/fe3b6ff5ad3a4df8b463168aac5185c73c2a50e3b0934c1482527da752bcca81?input:0>

3.3 利用区块链浏览器解析并学习区块链合约层构造

这一节我们利用以太坊区块链浏览器 Etherscan 学习智能合约的基本构造，高阶的技巧将在之后的合约实践课程中讲解。

首先，点击任意区块的详情，以 <https://etherscan.io/block/8413441> 为例，我们可以观察到合约层部署带来的一些不同：

Overview		Comments
① Block Height:	8413441	< >
② Timestamp:	④ 44 secs ago (Aug-24-2019 02:12:14 PM +UTC)	
③ Transactions:	159 transactions and 11 contract internal transactions in this block	
⑤ Mined by:	0x5a0b54d5dc17e0aad383d2db43b0a0d3e029c4c (Spark Pool) in 25 secs	
⑥ Block Reward:	2.10364150042983477 Ether (2 + 0.10364150042983477)	
⑦ Uncles Reward:	0	
⑧ Difficulty:	2,259,095,425,801,778	
⑨ Total Difficulty:	11,606,399,841,581,248,506,764	
⑩ Size:	26,459 bytes	
⑪ Gas Used:	7,994,010 (99.83%)	
⑫ Gas Limit:	8,007,795	
⑬ Extra Data:	PPYE sparkpool-eth-cn-hz3 (Hex:0x5050594520737061726b706f69c2d6574682d636e2d687a33)	
⑭ Hash:	0x9000e4c3100a65aa#05b3d509796d2e6b091cc71d3b0e9d825a59207b7a0883	
⑮ Parent Hash:	0x987fa08c9f6c7235e42a4e098b015794411a03d5169b1450220a9bda19b7f7	
⑯ Sha3Uncles:	0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a74130a142fd04d045347	
⑰ Nonce:	0xad12d5b006485176	
Click to see less ↑		

最为显著的一点,区块的大小不再有固定上限,而是由事务费用上限 **Gaslimit** 决定,而事务费用则直接与矿工所执行合约的总复杂度相关联。对于每一笔触发合约状态变动的事务,其复杂度都由所执行程序的指令加权每个指令的复杂度求和得到,而节点在发布事务前会附加相应的 **Gas** 并指定单位复杂度愿意支付的事务费用,矿工在执行该事务的过程中依次扣除执行费用,如果附加的 **Gas** 因不足被耗尽,则该次执行不会造成区块状态变动,且不会退回所消耗的费用。

可以理解的是,由于智能合约的编程语言是图灵完备的,事务的验证所消耗的算力不再可被忽略,为了避免庞大的计算开销以及恶意合约的影响,以太坊制定了以上的经济模型,限定每个区块所能处理的事务难度,这一举措也降低生态丧失去中心性的风险。

接下来,我们探究合约和其触发事务的状态,这里以著名的 ERC 代币合约为例(例如 ERC20, ERC621, ERC721),以太坊考虑到每个分布式应用需要形成自己的生态,甚至发行自己的代币,所以允许应用以代币合约的形式发行自己的代币,并与其他种类代币进行价值流通。定义此类功能需要遵循 ERC 合约规

范，并实现其规定的接口（以下为部分示例接口）：

```
TotalSupply //代币发行总量
```

```
BalanceOf (address _owner) constant returns (uint256 balance)//查询余额
```

```
transfer(address _to, uint256 _value) returns (bool success) //发送相应代币数目  
到钱包地址
```

应用开发者在此基础上再实现更加复杂的合约功能。以 CryptoKitties（一款用智能合约开发的虚拟宠物游戏）为例：

<https://etherscan.io/address/0x06012c8cf97bead5deae237070f9587f8e7a266d#code>

在其合约首页上便可以看到其代币的价值以及合约定义的所有方法，方便用户的学习、检测与调用，用户可以发布事务通过自己的账户或驱动其他合约与该合约交互。

Sponsored: YoBit.Net - Add your ERC20 Token to Yobit.Net Exchange!

Contract Overview CryptoKitties: Core

Balance: 65.384125562707918724 Ether

Ether Value: \$12,400.75 (@ \$189.66/ETH)

Token: \$4.38

More Info

My Name Tag: Not Available, login to update

Contract Creator: 0xba52c75764d6f59... at txn 0x691f348ef11e9e9...

Token Tracker: CryptoKitties (CK)

Transactions Internal Txns ERC20 Token Txns ERC721 Token Txns **Contract** Events Analytics Comments

Code Read Contract Write Contract

Contract Source Code Verified (Exact Match)

Contract Name: KittyCore

Compiler Version: v0.4.18+commit.9cf6e910

Optimization Enabled: Yes with 200 runs

Other Settings: default evmVersion

Contract Source Code (Solidity)

```
1- /**
2-  *Submitted for verification at Etherscan.io on 2017-11-28
3-  */
4-
5-  pragma solidity ^0.4.11;
6-
7-  /**
8-   * @title Ownable
9-   * @dev The Ownable contract has an owner address, and provides basic authorization control
10-   * functions, this simplifies the implementation of "user permissions".
11-   */
12-  contract Ownable {
13-    address public owner;
14-
15-
16-  }
```

在 events 一栏，可以看到该合约最近的状态变动，触发改变动的事务地址以及事务所调用的具体方法：

点击具体的事务地址可以获得关于该次状态变动更加详细的信息。

Etherscan 同样开放了 API 接口提供对于以太坊的状态获取:

但使用 API 需要申请相应的密钥，并受到每天 100 次访问的限制。

另外 Etherscan 收集了一系列有助于合约开发和学习的在线工具，有助于智

能合约的开发学习与漏洞检测:

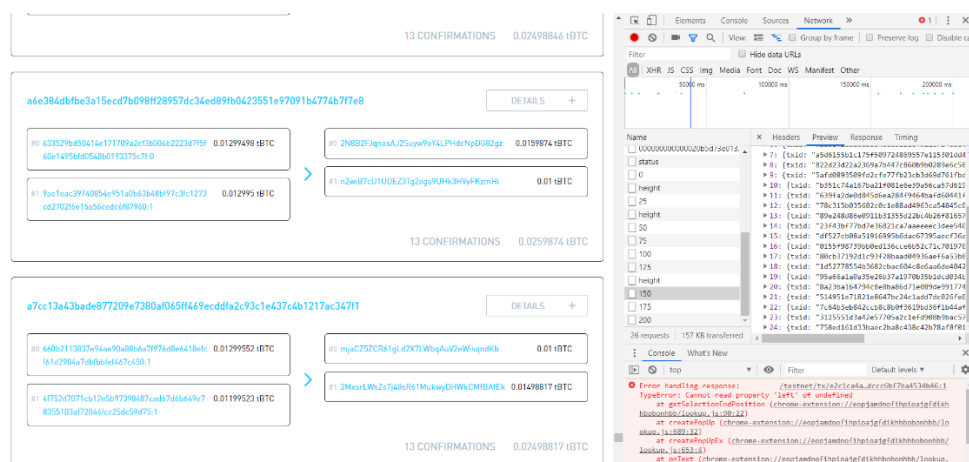
https://etherscan.io/directory/Smart_Contracts/Smart_Contracts_Audit_And_Security

3.4 拓展实验：批量获取并分析区块链元数据

以下实验为选作实验（加分项），不作强制要求，感兴趣的同学可以任选其一完成：

拓展实验 1：数据批量获取与挖掘-调用公开的 API 或使用爬虫获取今年 8 月期间的所有区块数据，对所有交易发布者所使用版本号 `version` 情况进行统计分析并作统计图，对 `locktime` 字段的使用比例进行统计分析，过滤出所有 `locktime` 不为 0 的交易，并将该交易数据集写入到一个文件内。

提示：爬虫的使用-利用程序模仿用户访问行为，根据 URL 所呈现的规律，对区块链浏览器返回数据进行批量过滤与提取，一般在 API 调用受限时才会考虑使用。



拓展实验 2：利用开源项目部署自己的区块链浏览器，并尝试调研区块链浏览器的构造方法。

提示：可用的开源项目-

<https://github.com/Blockstream/esplora>
<https://github.com/poanetwork/blockscout>
<https://github.com/carsenk/explorer>

五 实验报告要求

正常：简要记录实验流程（参考各个部分的练习）与相应的感悟体会；

有兴趣完成拓展实验的同学：无须完成以上内容，任选一扩展实验完成，并将详细的扩展探究过程整理并撰写到实验报告上。