## 动态可变映射哈希区间

我：认为只有出现新路径才会出现漏洞，并且设定的是在目标哈希区间内才认为有新路径，不在目标区间内就认为它没有新路径

师：它不是严格一一对应的，哈希区间的分布不是均匀的，所以目标哈希区间和新路径正常来讲是有了新路径就可以，但有时候计算出的hash值是落不到目标区间中的，所以目标区间变成动态变化的。当跑一段时间发现都没有落在这个部分，就把这个区间变大。

我：那一开始该怎么设定目标哈希区间呢

师：刚开始是肯定有新路径的，那hash区间就是全部的区间，[0, 2^256 - 1]，后面就是对区间缩减，这里的调整参数α，可以先设置为0.5，缩到后面就不在区间里面，不在里面运行一段时间就没结果，就可以动态把窗口变大。

我：对标比特币的话，挖矿难度是具体清晰的，这里的话增大难度好像没有一个定量描述的东西

师：这里的难度增大可以类似于在长时间里没有相应的新路径，原先的hash计算的算力师固定的，所以比特币是通过前导0的个数来确定难度的，现在是反过来，每一次挖到新路径是不固定的，所以区间就变成了动态调整的。

我：我们这里执行一次的时间也是固定的，是执行窗口的时间

师：比特币是没有动态窗口的概念，它只是说。。。？咱们里面虽然说是有一个单次的窗口，但实际上难度你是不知道的

我：那因为我们这个窗口每次计算的值也是不一样的，那能不能直接用比特币的那种方法

师：也可以啊，但问题是这样对比不到你的新路径，**这里的困难是咱们这个区间和新路径的hash的映射关系是不清楚的**。所以这里面的难度可以想象成就是说，为什么我们引入动态区间的概念，就是相当于说这个**难度是由动态区间标注的**，正是因为我们不知道新路径和区间的映射关系，所以刚开始设置区间很大，大家挖矿都很/； 快，完了之后缩减，到一定时间后很长时间都不会出新的，然后这个区间又会扩大，所以这个难度是由区间控制的，但实际上每一次在窗口里计算的东西是和比特币计算的是一样的

我：那就是一开始的话，因为范围特别大，所以出块时间特别快

师：对，最终的期望是在固定时间块内有人一定获得记账权，相当于是调整一段时间后达成这样的期望，但不是绝对的，最初的记账时间很快，后面会很慢，但期望值是固定的。

比特币会每隔固定的时间调整难度

第一个获得之后区间之间减半，第二个再减半。。。。隔一段时间之后就没有落在区间里的，再扩大。什么时候能对标到比特币的减半呢？每次减半后都计算前一段时间的平均期望，看是不是到达预期值。比如执行50次窗口，前面几次因为区间太大了肯定在，但是后面的话不一定每次窗口都会成功挖矿。α应该是一个通过拟合之后的参数，现在先这么取0.5,0.6,0.7…这种线性的

新路径是优先条件，有新路径的时候是不看hash值有没有落进区间的，动态的记账权是没有新路径的时候

|  |  |  |
| --- | --- | --- |
|  | Proof-of-work | Proof-of-fuzzing |
| 计算区块hash值 | Hash(preHash, version, timestamp, difficulty, nonce, Merkle Root) | Hash(preHash, version, timestamp, difficulty, List<Payload(input,path,iscrash)>,nonce Merkle Root) |
| 联系：pof中的List+nonce相当于pow中的nonce | |
| 挖矿成功条件 | 该区块Hash第一个满足前n位都为0 | 优先级：   1. 新路径产生（一开始新路径肯定会特别多，一个执行窗口就挖矿成功，出块时间太短，导致短时间内很多的节点同时算出了目标值得到了记账权，这样就产生了很多分叉，这对系统的安全性没有好处） 2. 如果没有新路径的话，该区块Hash在某个区间范围中   ↓  若执行窗口有新路径，则最快探索到新路径(每条新路径记录时间戳)以及窗口hash落在区间内取胜；若一直没有新路径，则就看hash是否满足要求  一次窗口中如果有多条路径，计算每条的Hash，  新路径判定方法 |
| 挖矿终止 | / | 代码覆盖率稳定，测试接近饱和，一直没有新路径被发现；  矿工主动退出 |
| 难度设定 | 一次执行hash计算的时间和资源消耗是差不多的，经过非常多次的计算来寻找前n位都为0的Hash值，n越大难度越高（n动态变化） | 寻找落在某区间内的Hash值，区间越小难度越高（区间动态变化）  区别：一次执行一条路径的时间和资源消耗不尽相同，可能相差很大，为对标pow中一次hash时间基本相同，设定一个固定时间的执行窗口 |
| 联系：pow执行一次hash计算——pof执行一次固定时间窗口  Pow多次hash以寻找前n位为0的hash——pof多次执行时间窗口以寻找落在区间内的hash | |
| 挖矿速度 | 每10分钟出一个区块 | 每5分钟出一个区块（要考虑到这是一个寻找漏洞的赏金计划，所以不能时间太久 ） |
| 难度调整方案 | 每2016个区块（大约14天）调整一次挖矿难度，公式为： New Difficulty = Old Difficulty \* (Actual Time of Last 2016 Blocks / 20160 minutes) | 期望难度因子α来调整区间范围  **期望调整α后达到落在区间中才有新路径的最终效果(不好实现)→期望调整α后达到**  新路径—目标区间（放弃关联二者）  每轮挖矿都取全hash集[0,2^256-1]中的随机区间，动态调整的是区间长度，即区间头为随机值，加上区间长度，区间长度由α决定  在一次执行窗口之后（对标pow计算完一次hash），统计新路径的个数 |
| 校验提交的区块合法性 |  | 1. 验证是不是新路径（需要存储所有的新路径？） |

##### 为什么要调整比特币的Mining难度？

如果不调整难度，由于人数的增加，系统中的算力会变得越来越强，在这种情况下难度保持不变的话，出块时间会越来越短。当出块时间缩短到比区块在网络上的传播时间短的时候，由于网络的延迟，就会出现节点不能及时收到出块的消息还在沿着之前的区块链扩展的情况，如果这时有两个节点同时发布一个区块，就会出现分岔。随着出块时间越来越短，分岔越来越多，就会将系统的算力分散，这时就危害到了系统的安全性。因为这时恶意的节点不需要掌握51%的算力就可以攻击系统了，比如分岔攻击。

##### 比特币为什么设定10分钟出一个区块

其实这是中本聪(比特币的发明者)自己决定的,区块的生产速率应保持每10分钟1个区块的平均平均速度,因为:**区块链估计需要10分钟才能将最新的区块传播到全球所有节点**。

比特币平均出块时间(Block interval )为 10minutes，是难度系数与全网算力共同作用的一个动态结果。所以这个时间并不是一个确定的值，通过[http://Blockchain.info](https://link.zhihu.com/?target=http%3A//Blockchain.info)上的统计我们可以看到时间是在10分钟左右变化的。

##### 比特币的算力受什么影响？

矿机性能、矿工数量、矿池规模

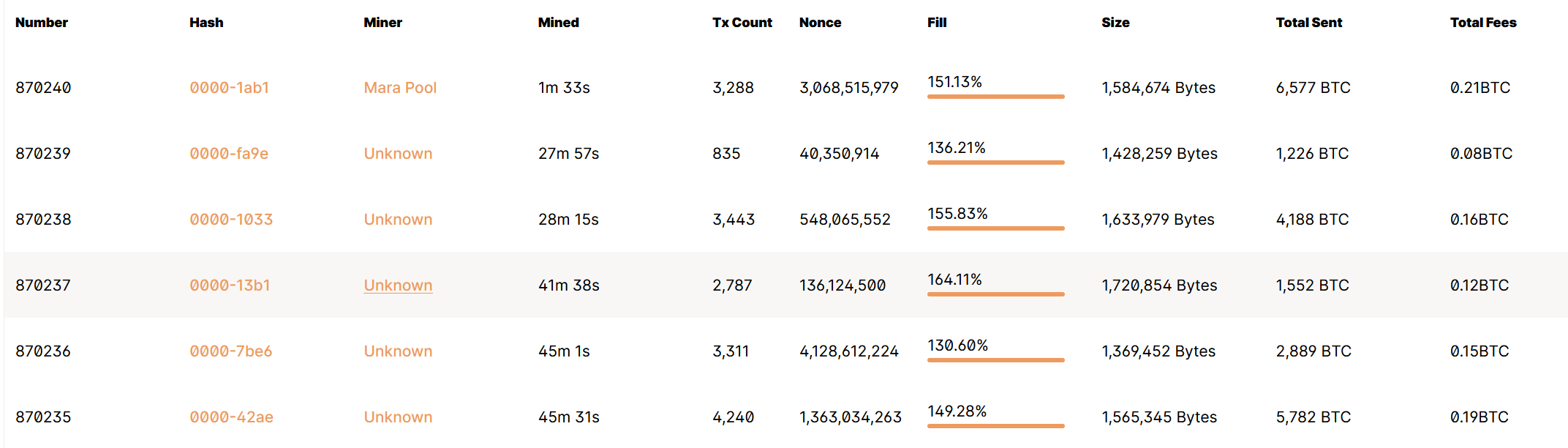
**挖矿算法**

在256位hash空间中，hash值的范围是[0，2^256 - 1]，记每次计算hash的时间为1s（即窗口时间，窗口结束后的hash计算可忽略不计）

**理论假设**

Hash全集的长度为，设目标出块时间为，动态区间长度为，我们认为sha-256计算出的hash值在统计学上是均匀分布的，则命中的概率是，平均需要次计算才能命中区间，也就是说要s挖矿成功一次, 即

预期：所有运行次数(y)和命中次数(x)的关系为

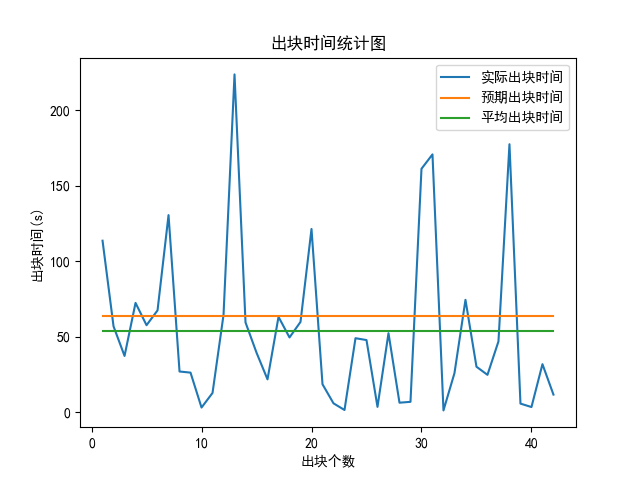
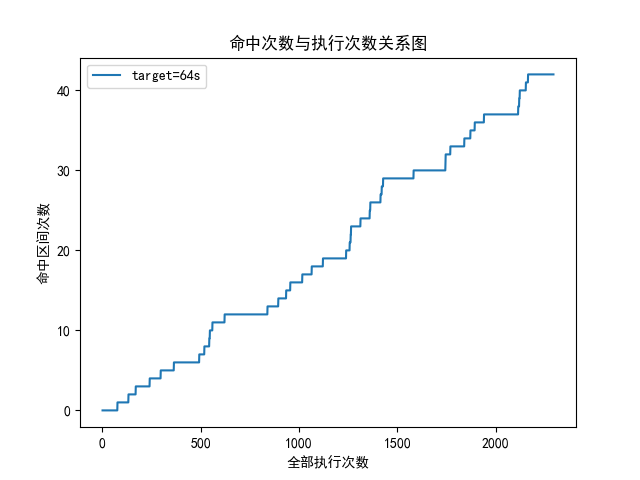


**实验**

区间长度为，预期出块时间越长，越小，区间越短，挖矿难度越大。区间头为随机生成。

**实验1：**

预期出块时间：（即64个窗口出块，暂不考虑延时等因素），理论上，计算，得出，即区间长度为



区间范围：[44,1809251394333065553493296640760748560207343510400633813116524750123642650668]

执行了2294次窗口，命中次数为42

平均出块时间：53.908261904761915

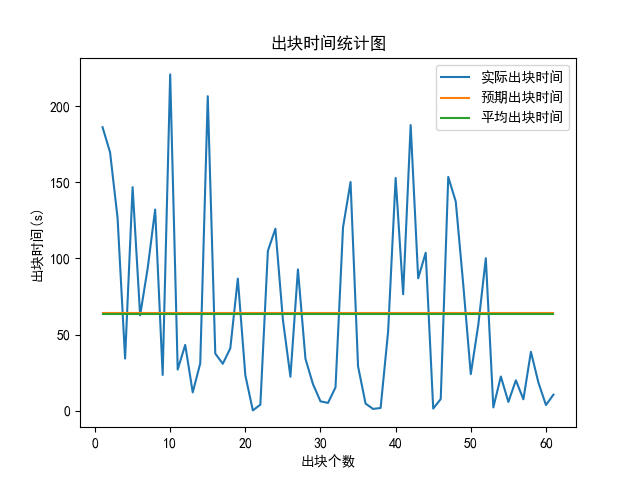
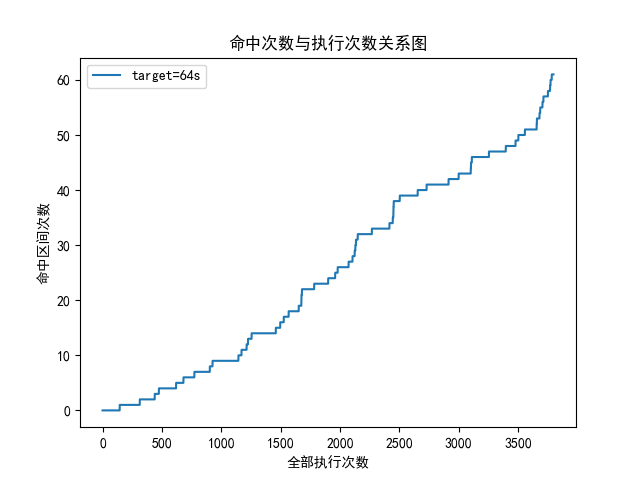
全部出块时间：[113.589 56.859 37.256 72.398 57.623 67.439 130.545 26.944 26.202

3.076 12.737 64.421 223.756 59.454 39.432 21.742 63.148 49.506

59.794 121.381 18.503 5.863 1.451 49.006 47.776 3.518 52.318

6.281 6.856 161.234 170.772 1.18 25.877 74.421 30.112 24.739

46.905 177.502 5.679 3.4 31.802 11.65 ]



区间范围：[54,1809251394333065553493296640760748560207343510400633813116524750123642650678]

执行了3798次窗口，命中次数是61

平均出块时间：63.50068852459017

全部出块时间：[1.86090e+02 1.69610e+02 1.26444e+02 3.42210e+01 1.46738e+02 6.25750e+01

9.33470e+01 1.32033e+02 2.34520e+01 2.20759e+02 2.70150e+01 4.31850e+01

1.19780e+01 3.10420e+01 2.06431e+02 3.74900e+01 3.07810e+01 4.09390e+01

8.67210e+01 2.32180e+01 1.74000e-01 3.94000e+00 1.04904e+02 1.19472e+02

5.96850e+01 2.22640e+01 9.27530e+01 3.40770e+01 1.74270e+01 6.11700e+00

5.10200e+00 1.53020e+01 1.20369e+02 1.50119e+02 2.90890e+01 4.63500e+00

1.13700e+00 1.75800e+00 5.20050e+01 1.52785e+02 7.64180e+01 1.87474e+02

8.69170e+01 1.03668e+02 1.37600e+00 7.62600e+00 1.53471e+02 1.37290e+02

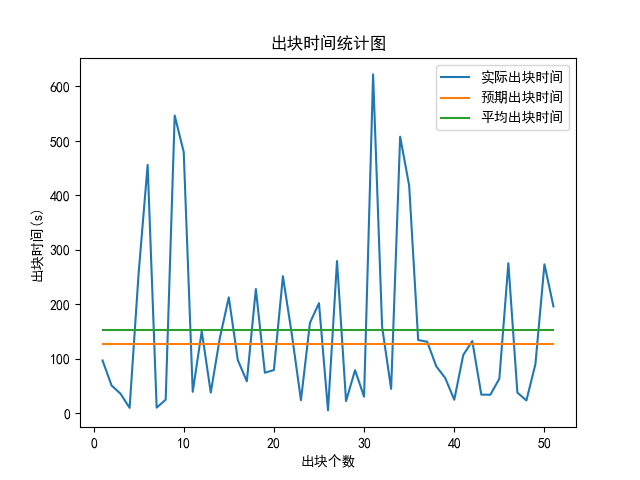
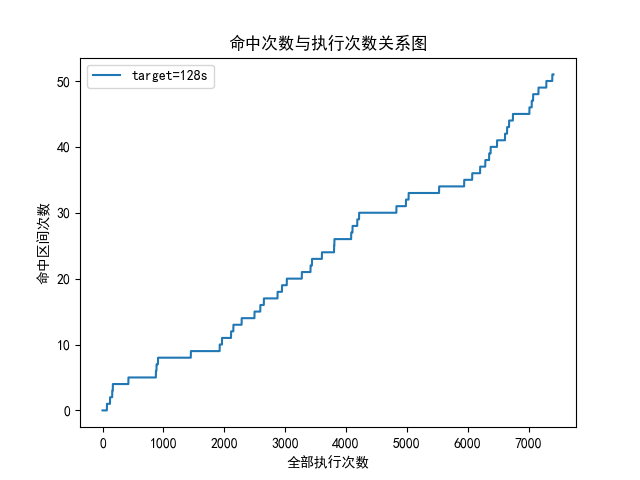
8.26670e+01 2.39840e+01 5.63990e+01 1.00108e+02 2.13700e+00 2.24240e+01

5.73400e+00 1.99270e+01 7.45800e+00 3.86460e+01 1.84320e+01 3.69900e+00

1.05040e+01]

**实验2：**

目标出块时间：，则，区间长度



区间范围：[98,904625697166532776746648320380374280103671755200316906558262375061821325410]

执行了7410次窗口，命中次数为51

平均出块时间： 151.9302156862745

全部出块时间：[ 96.882 50.995 35.656 9.835 256.807 456.047 10.25 24.877 546.617

479.784 39.317 150.589 37.92 137.627 212.799 97.533 58.739 228.074

74.387 79.316 251.605 143.771 23.878 165.937 202.187 5.216 279.524

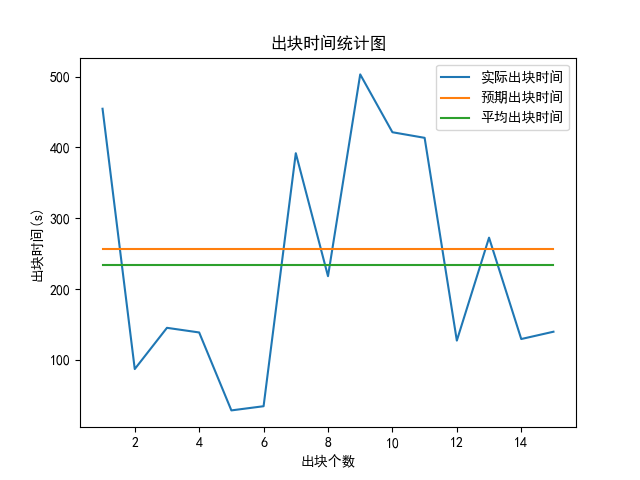
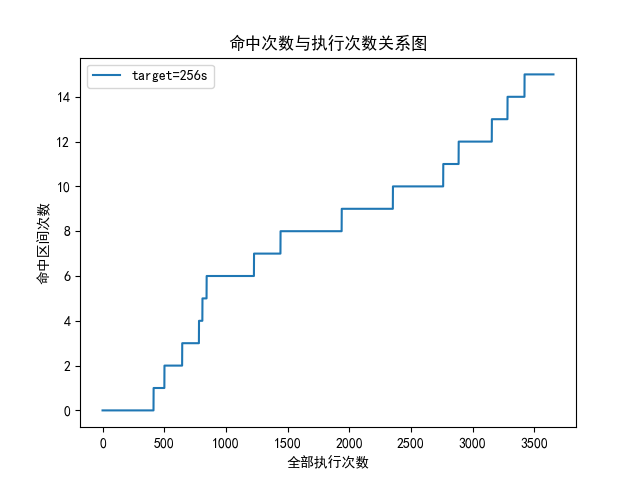
22.297 79.039 30.399 622.174 158.62 44.716 507.873 417.952 134.434

131.268 86.371 64.609 24.71 106.997 132.5 34.192 33.929 63.419

275.476 37.918 23.482 90.245 273.489 196.163]

**实验3：**

目标出块时间：，则，区间长度



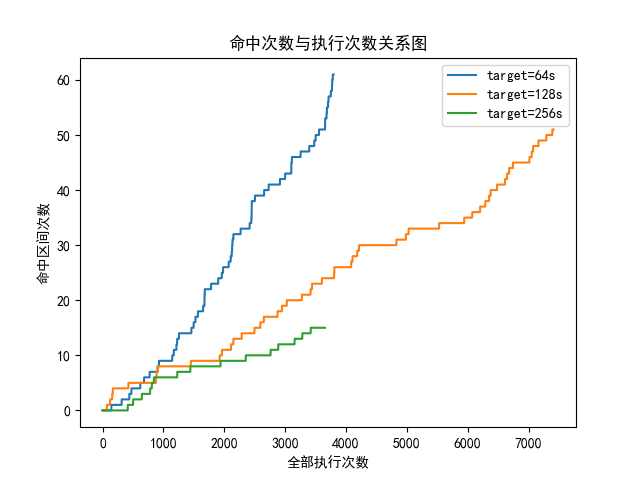
区间范围：[211,452312848583266388373324160190187140051835877600158453279131187530910662867]

执行了3655次窗口，命中次数为15

平均出块时间： 233.66133333333335

全部出块时间：[454.65 86.898 145.145 138.594 28.488 34.397 391.857 218.098 503.175

421.465 413.559 127.099 272.526 129.323 139.646]

****