链滴

bitcoin 与工作量证明

作者：shooter

原文链接：https://ld246.com/article/1532584129262

来源网站：链滴

来自简书

btc address: 1FmWXNJT3jVKaHBQs2gAs6PLGVWx1zPPHf

挖矿的时候需要经常听到一个词 **工作量证明** , 即Proof Of Work,简称POW.

POW是个什么鬼, 我们先抛开这个概念, 玩个游戏.

假如有很多人喜欢问我很八卦的问题, 但我 只想答有限的几个,
那让谁答呢? 我出个游戏规则:

1 先把问题列出来, 重复的问题我不回答啊
你有异装癖么

2 做hash运算, 就做sha256吧

```
require 'Digest'
Digest::SHA256.hexdigest("你有异装癖么")
```

得到4f65c9ca420e5f191d9f2730c4f99c7fe8f2301a431bade918de9d38f7401a4e

3 计算hash值, 这个难度太低了, 身边的一堆程序猿都会算.

我希望 你的问题 加上一个自然数,合并成一个新的字符串,再做hash运算, 得到新的hash值, 使得新的hhs值前4个都是0, 并且这个自然数是最小的, 那好,我就回答你的问题.

一言不合就撸码

```
require 'Digest'

10000.times do |n|
 issue = "你有异装癖么"
 val = issue + n.to_s

 ret = Digest::SHA256.hexdigest(val)

 if ret[0..3] == "0000" #判断hash值的前4位是0么
   puts val
   puts n
   puts ret
   break
 end

end

#你有异装癖么3699
#3699
#00007ed47ec0280ca933ed5dc9396892fee27c9dbec9c07f95f0247169df1bee
```

得到 3699,

Digest::SHA256.hexdigest("你有异装癖么3699")的结果符合前4位是0.

你把问题 你有异装癖么3699给我, 我通过验证, 就回答这个问题.

你有疑惑了, 为啥要从 [1,10000] 这个区间顺序计算呢?

不能随便从一个数字算么?

因为hash运算的结果是无序的, 而且不能进行逆运算, 顺序计算比较方便, 没有遗漏. 通过这个方式得的自然数, 也是符合游戏规则的最小自然数, 因为是顺序计算的啊

嗯的, 在回到这个游戏, 大家都遵守这个规则, 我也在回答这个问题,

又过了一段儿时间, 需要回答的八卦问题还是太多了, 我吃不消了,囧.

我定个新规则, 其他的不变, hash值前5位是0的, 我才回答.

然后继续运行, 到了我又吃不消的时候, 继续改, 前6位, 前7位, 前8位,

前n位是0.

如果你继续算下去, 你会发现, 算的时间会越来越长,我们可以说难度增加了. 有这样直观的感觉, 前几是0的个数越多, 这个难度越大, 计算时间越长.

难度到底有多大, 也就是时间到底有多长呢?

我拿自己的 2014年款的 mac pro中款粗略算了下,语言是ruby,

时间就是金钱, 这个时间还是比较可观的.

```
Digest::SHA256.hexdigest("shooter36")      #04b4ee0f1b56950f1f9880d076b7449c66705d571a939a36a49d9973dd50ab3    2ms
Digest::SHA256.hexdigest("shooter578")     #002bae033279e48835e9a3c8b71a6835bf171a20b700a10ba4bd63710bfcb49    15ms
Digest::SHA256.hexdigest("shooter4434")    #00072b50cc7963a310962af33efcd5109cdfb6ac63ce6bb275ce243f3ec247b    406ms
Digest::SHA256.hexdigest("shooter35786")   #00002d8c31f6eb29d848cdf520b499cd9f729b1dd037275d82935b7766eaa3e    4309ms
Digest::SHA256.hexdigest("shooter717095")  #00000178662b978d8d7cf930a1caceae70f09734b4fe6423eef2a45ccfb9ce7    82468ms
Digest::SHA256.hexdigest("shooter2038373") #0000001ef511d494ecdddcd2abfded0a34df87ac76410452e01e0a2958bf0a5    272266ms
```

说了这么多, 这个游戏规则其实就是POW.

用到bitcoin上,无非是要替换些东西, 八卦问题 => bitcoin需要的数据,

替换bitcoin需要的难度. 理解这个思想很重要, 别的细节慢慢聊.

bitcoin系统的难度有多大呢?

块高度477,360 的hash 000000000000000000e8af89716e0ea8a2aa7c7b789935e98f0a32a4a31247f有18个0.

Nonce 是十六进制的0x71eee8ca,即算了 1911482570次, 可以试试用一般的笔记本计算19亿次has需要多长时间.

POW有什么特点呢?

1 运算不可逆, 就代表谁也不可能偷懒

2 每个值的hash都不一样, 意味着不同的问题都要算, 换个新问题, 就要重新算.

3 你算起来很费劲, 花费时间长

4 我验证起来很简单, 把结果hash一下, 符合规则的我就回答, 不符合的抛弃

5 这个规则实现还是比较简单的

结果就是 遵守游戏规则的人很艰难的算出一个结果,会很珍惜,因为你付出了很多的计算资源跟时间, 你望我能认可你的努力.

这个概念在bitcoin系统中很重要

参考:

https://www.bilibili.com/video/av12465079/

https://en.bitcoin.it/wiki/Proof_of_work