B锁链 密码

灵宗, 博士
高级软件工程师/科学家
ibm almaden 研究中心
美国加利福尼亚州圣何塞

哈希评论

我们定义了一个加密哈希函数: $h:\{0,1\}^{*} \longmapsto \{0,1\}^{K}$

将一些任意大小的位字符串映射到一些固定大小的位字符串。

该函数是确定性的;**相同的输入总是产生相同的输出**.

"现代密码学的工作马"-布鲁斯·施尔

示例: sha256 映射到256位字符串

"你好，世界！"沙其6

0xd9014c462444a5bac3147776b689ad467fa44a1a1a1a9b8a9d55f72ff5

加密哈希函数的性质

符号：■是隐藏的■ 是公开的

预映像电阻:

让X= {0, 1}*=消息

Y=H(X)

X= h^{-1}(Y) →应该是计算上困难的

+ 找到哈希输出的预置图像 (原始值) 应该是极其困难的。

二次预映像电阻:

给定的消息X

找到一些x 's. t。 H(x ') =H(X) 应该是计算困难的。

加密哈希函数的性质

抗碰撞性:

假设我们有两条消息X_1,X_2

它们的哈希值相等的概率, p ($H(X_1) = H(X_2)$), 应该是 "非常小的"

同样, 找到X_1,X_2这样的$H(X_1) = H(X_2)$ 应该是计算困难的。

查找碰撞的上限最多是 o ($n^{半}$) (称为生日攻击)

为什么加密哈希函数有用?

就比特币和其他加密货币的上下文:

+ 默克树
+ 工作证明 (比特币、拉泰科因、狗币等)
+ 事务, 块, 地址都引用哈希值

也

+ 许多密码协议中的基本操作
+ 基于哈希的消息身份验证代码 (hmac)
+ 密码验证
+ 承诺计划
+ 伪随机数发生器 (prng)

简单的哈希承诺计划

为什么这些哈希属性有用?

想一个简单的例子: 爱丽丝和鲍勃在硬币翻转上赌100美元

1) 爱丽丝称硬币翻转的结果
2) 鲍勃翻转硬币
3) 爱丽丝赢了100美元, 如果她的猜测是正确的

现在, 如果爱丽丝和鲍勃分开, 不信任对方呢?

- 爱丽丝想给鲍勃一个*承诺*她的猜测, 没有透露她的猜测, 鲍勃翻转硬币之前, 否则鲍勃可以作弊!

简单的哈希承诺计划

相反, 我们可以修改我们的 "协议", 以约束爱丽丝的猜测与承诺

1) 爱丽丝选择了一个很大的随机数R.
2) 爱丽丝猜出硬币翻转的结果,B.
3) 爱丽丝生成一个*承诺*硬币翻转,C=H(B||R)
4) 爱丽丝把这个承诺寄给鲍勃。
5) 鲍勃翻转硬币, 并将价值发送给爱丽丝。
6) 爱丽丝送鲍勃的随机数和她的猜测: (r ',b ')
7) 鲍勃然后检查c '=H(r '||b ') =C=H(B||R), 以确保爱丽丝没有改变她的猜测中期承诺。
8) 两人现在都可以就谁赢了 1 0 0 元达成一致。

简单的哈希承诺计划-作弊

鲍勃怎么能骗爱丽丝?

1) 当鲍勃收到C=H(B||R), 如果他能计算 $h^{-1}(C) = B||R$, 鲍勃可以恢复爱丽丝的猜测, 并给她相反的结果!

如果我们的哈希函数H是**抗预映像**, 这不应该是可能的。

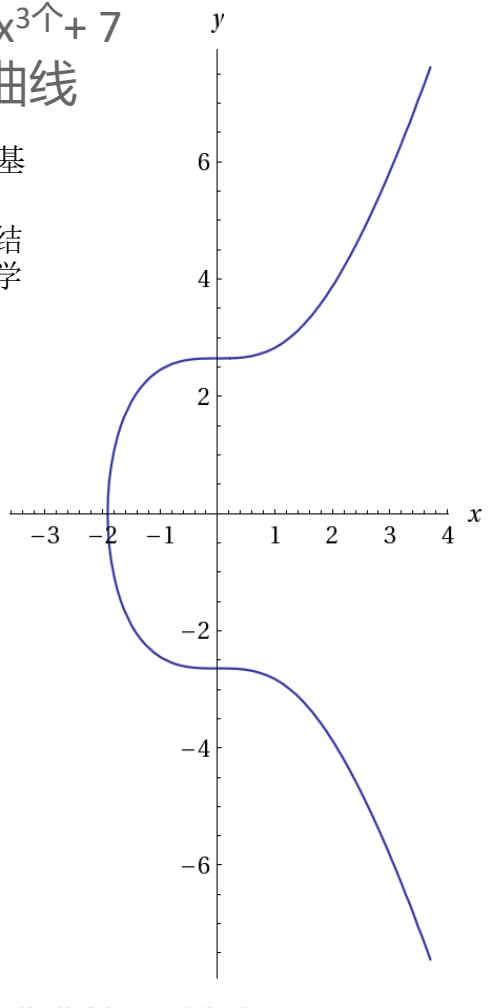
爱丽丝怎么能骗鲍勃?

1) 爱丽丝把她的承诺寄给鲍勃C=H(B||R), 但揭示了相反的猜测, (!B,r ').爱丽丝赢了, 如果她能挑r 's. t。 c '=H(!B||r ') =C.

如果我们的哈希函数, 这将失败,H是**第二预映像电阻**!

椭圆曲线

赛文256k1:$y^2 = x^{3个} + 7$
比特币的椭圆曲线



椭圆曲线密码学(Ecc) 是一种基于有限域椭圆曲线代数结构的公钥密码学方法。与非 ec 加密技术 (基于普通伽罗瓦字段) 相比, ecc 需要更小的密钥来提供等效的安全性。
椭圆曲线适用于密钥协议、数字签名、伪随机生成器等任务。通过将密钥协议与对称加密方案相结合, 可以间接地将它们用于加密。它们还用于几种基于椭圆曲线的整数分解算法, 这些曲线在密码学中具有应用, 例如 lenstra 椭圆曲线分解。

椭圆曲线由以下仿射长的 weierstrass 形式定义:

$$E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$$

我们通常考虑短的 weierstrass 形式:

$$E : Y^2 = X^3 + aX + b$$

在大多数情况下, 您需要了解的椭圆曲线是它们提供了另一个具有某些所需属性的有限阿贝尔群。

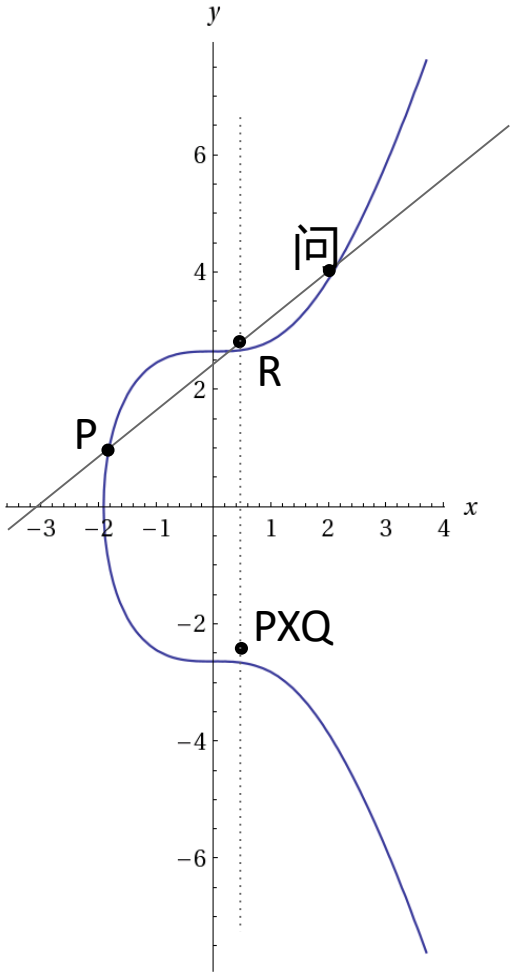
https://en.wikipedia.org/wiki/Elliptic-curve_cryptography

椭圆曲线-群法

可以使用弦切线过程定义椭圆曲线上的群律。

给出两个椭圆曲线点, p 和 q, 我们定义为 pxq 如下:

> 我们找到了相交 p 和 q 的线, 它必须与最后一点, r 相交。如果我们然后在 x 轴上反射 r, 我们得到另一个点, 我们定义为 pxq。



椭圆曲线-群法

更正式的是

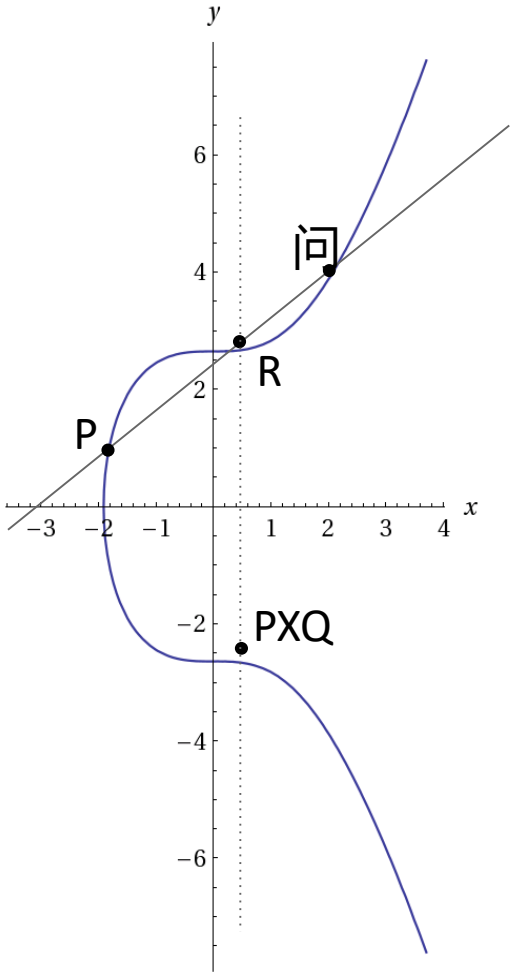
$$\text{Let } P = (x_1, y_1),\ Q = (x_2, y_2),\ P \times Q = (x_3, y_3)$$

$$s = \frac{y_2 - y_1}{x_2 - x_1}$$

$$x_3 = s^2 - x_1 - x_2$$

$$y_3 = s(x_1 - x_2) - y_1$$

在某些曲线和有限域上, 这形成一个循环 (或几乎是循环的) 有限阿贝尔群。



椭圆曲线-群法

如果 p = q, 我们在 p 处找到切线, 将其扩展到点, r, 然后在 x 轴上反射到 p^2.

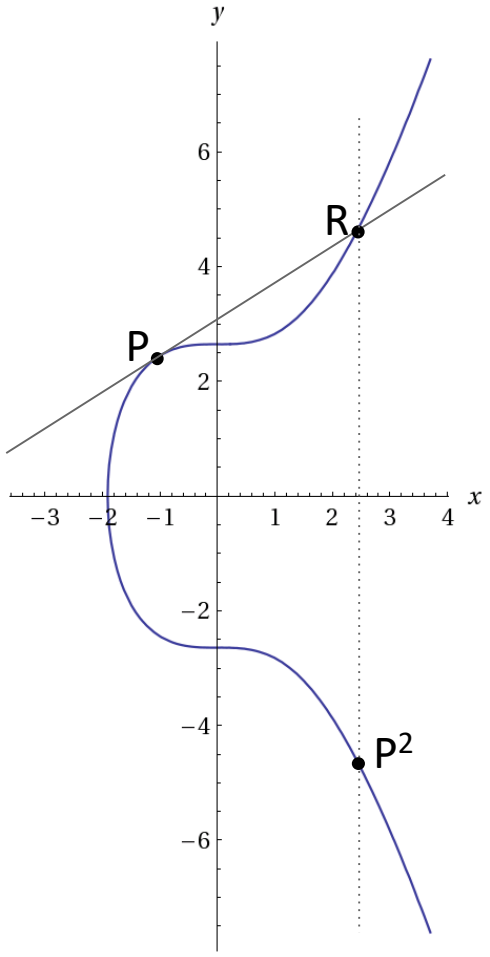
$$P = (x, y)$$

$$s = \frac{3x^2 - a}{2y}$$

$$x' = s^2 - 2x$$

$$y' = s(x - x') - y$$

$$P^2 = (x', y')$$



椭圆曲线离散对数问题 (ecdlp)

对于某些正整数 m, 我们可以定义:

$$P^m = P \times P \times \cdots \times P = \prod_{i=1}^{m} P$$

相信在 p 和 p 的情况下, 发现我米

$$m = \log_P P^m$$

在某些有限域和某些曲线上计算上是困难的。

因此, ecdlp 构成了椭圆曲线密码学的基础。

ecdlp

与香草有限场上的离散对数相比,**椭圆曲线离散对数问题没有已知的亚指数算法**(假设曲线不是超标或异常的)。

椭圆曲线离散对数的最快、最实用的算法是**平行波拉德的罗**, 在 o (n) 中运行半).

为了实现相当于128位块密码的安全性, 我们需要选择一条群阶曲线2^{256}.

将其与 rsa 或其他基于因子的算法进行比较, 这需要2048位密钥来实现等效的安全性。

数字签名方案 (dss)

类似于手写签名。

其他人可以验证带有您签名的邮件实际上是由您编写的。

同样, 没有你也很难伪造签名。

示例：使用我的 gpg 密钥对文本文件进行签名

"你好，世界!"

```
owebtqky\ zanawacaroxygnahi6..。qrdg\ b0krumnobdf7ht b9wtg =
```

dss 安全定义

(邮件、 sig) 收件人需要以下属性:

+ **消息完整性**-消息在发送和接收之间未被修改。
+ **消息来源**-该邮件确实是由原始发送方发送的。
+ **不可否认性**-原发件人不能回溯并声称他们没有发送邮件。

数字签名方案

更正式地说, 数字签名方案由两种算法组成:

+ 一个签名算法,标志, 它使用了一个密钥,Sk.

  s=标志(米,Sk),s是某些消息的签名,米.

+ 一种验证算法,验证, 它使用公钥,Pk.

  有效=验证(标志(米,Sk),Pk),无效=验证(s ',Pk) 为所有的 '! 的。

椭圆曲线数字签名算法

ecdsa 的定义如下:

e: 椭圆曲线。

G: 具有较大素数顺序的椭圆曲线的生成器点,P.

P: 一个大的, 素数整数, 其中G^P= o。

H: 加密哈希函数。

ecdsa-设置

签名者创建:

秘密钥匙Sk, 从 [0,...,..,P-1]。

公钥,$Pk=G^{Sk}$, 应公开分发。

ecdsa-标志

标志(米,Sk):

H=H(米) 哈希消息

Z=H[0: 日志$_2$P] 获取日志$_2$ P最左侧的位H

K= 从 [1,...,P-1]K必须保密!

R= x-coord (G^K) (modP) x-coord (p = (x, y)) = x

s= (Z+Sk* *R) *K^{-1}(modP)

返回 (R,s)**我们的邮件的签名**

ecdsa-验证

验证(R,s,米,Pk):

$Z=H(米)$ [0,..., 日志$_2$ P]

$a=Z^* *s^{-1}(\text{mod}P)$

$B=R^* *s^{-1}(\text{mod}P)$

$V=$ x-coord $(G^{a*} *Pk^B)$ (modP)

返回有效如果VR(modP)否则无效

ecdsa-快速证明草图

让$w=s^{-1}$

$G^{a}* *Pk^{B}=G^{Z}* *w* *G^{Sk}* *R* *w$

$=G^{Z}* *w+Sk* *R* *w$

$=G^{(Z+Sk* *R)} *w$

$=G^{(Z+Sk* *R)} * (Z+Sk* *R^ (-1) *K^ (-1) ^ (-1)$

$=G^{K}$

$V= x\text{-coord} (G^{a}* *Pk^{B}) (mod P)$

$= x\text{-coord} (G^{K}) (mod P)$

$=R(mod P)$

在比特币中使用 ecdsa 签名来证明交易输出的所有权!

基于会计与基于事务的分类帐

基于会计

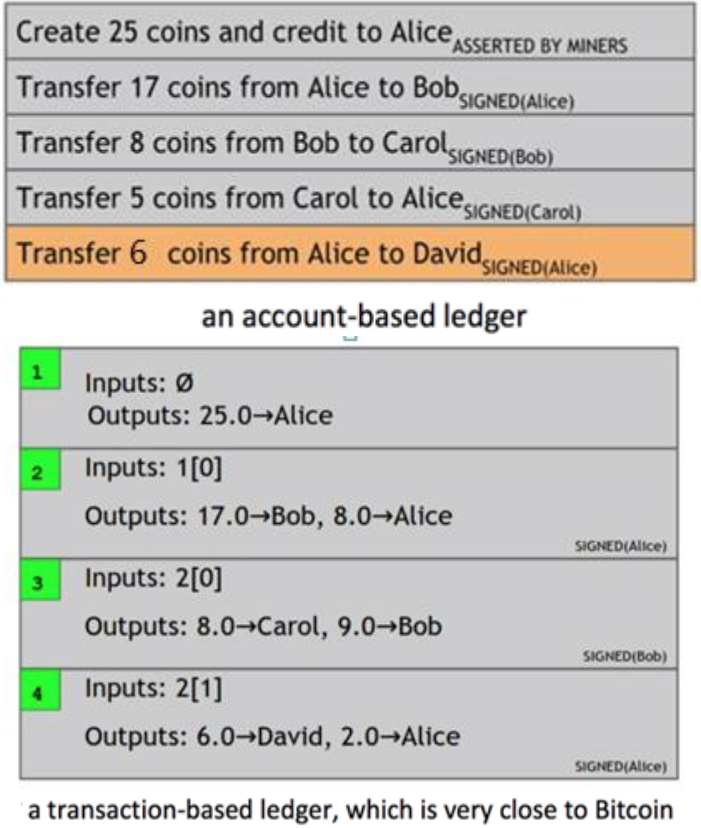
- 必须跟踪影响 alice 的每笔交易
- 需要额外的维护, 容易出错

比特币是基于交易的分类帐 (三重输入记帐)。

特征:

- 更改地址-必需, 因为 tx 输出仅花费一次
- 高效验证-只读最新历史
- 联合付款-alice + bob 表格 1 tx

(信用为内容组织和图去普林斯顿课本)



a 的内容
比特币交易

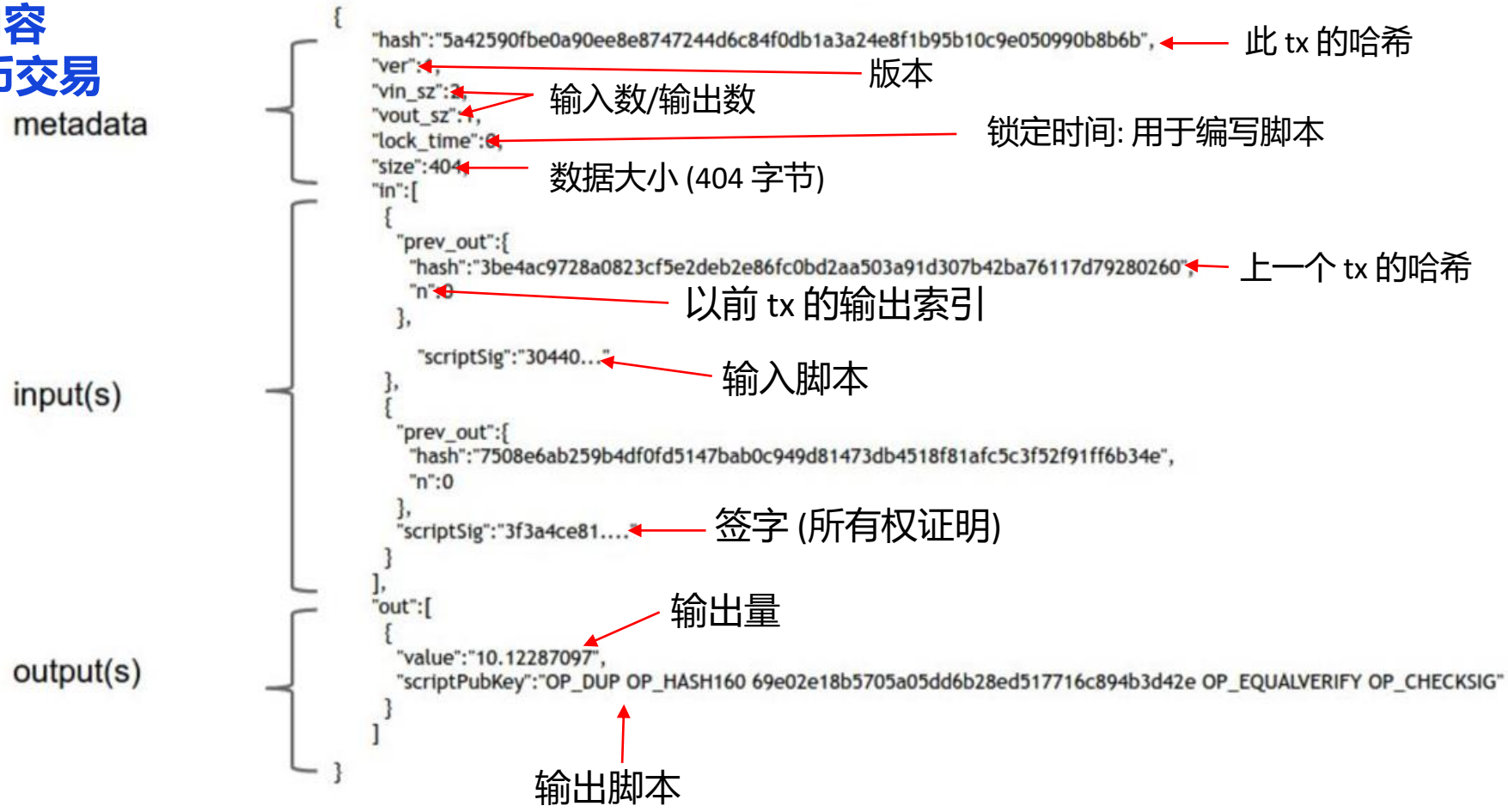
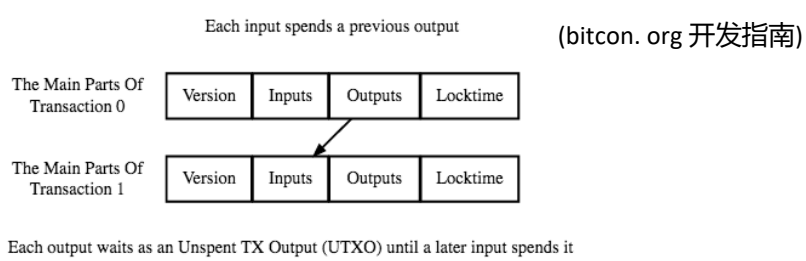


Figure 3.3 **An actual Bitcoin transaction.**

比特币脚本

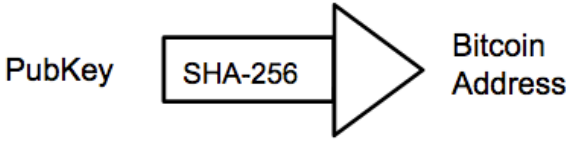


(bitcon. org 开发指南)

输出 "地址" 是真正的脚本。

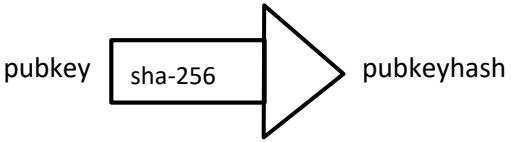
通过脚本输入和输出允许
比特币的未来可扩展性。

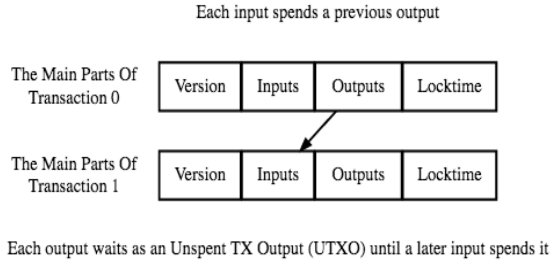
- 记得：
  - **签名**证明公钥的所有权
  - **比特币地址** = = **哈希 (pubkey)**
  - 由输入和输出组成的事务**地址**
  - 花费比特币是**赎回**以前的交易记录输出
- "此金额可由**签名**从地址 x 的所有者 "
  - 无法从地址中获取公钥!
- "这一金额可由**公钥**哈希地址为 x, 再加上**签名**从该公钥的所有者 "



(普林斯顿教科书中的语录)

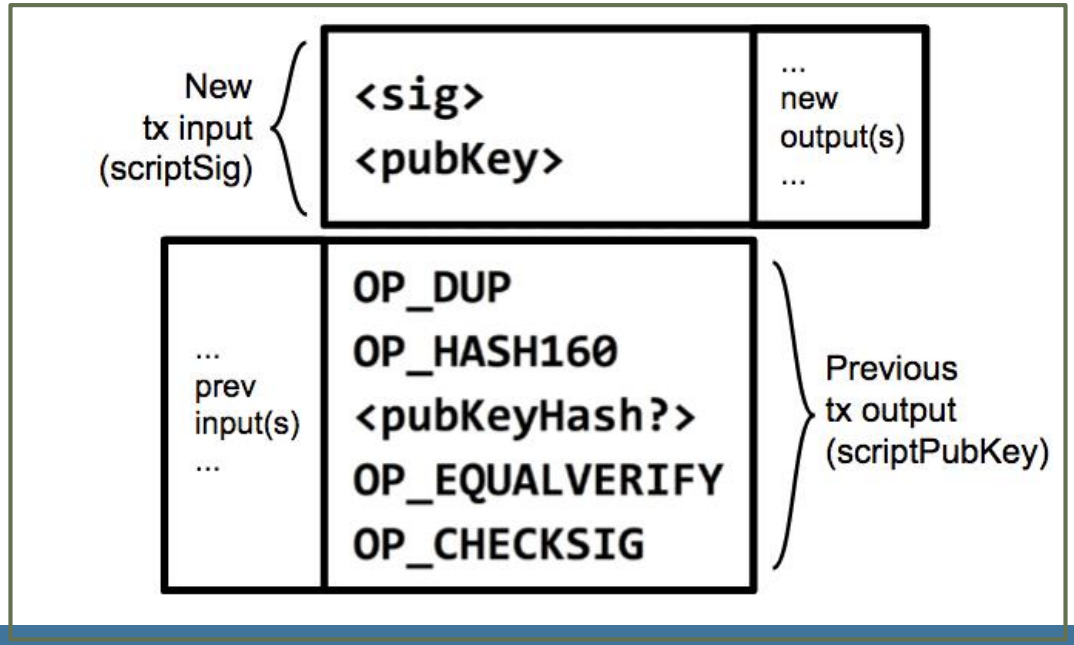
比特币脚本





记住: 哈希 (pubkey) = = 地址 = = "pubkeyhash"

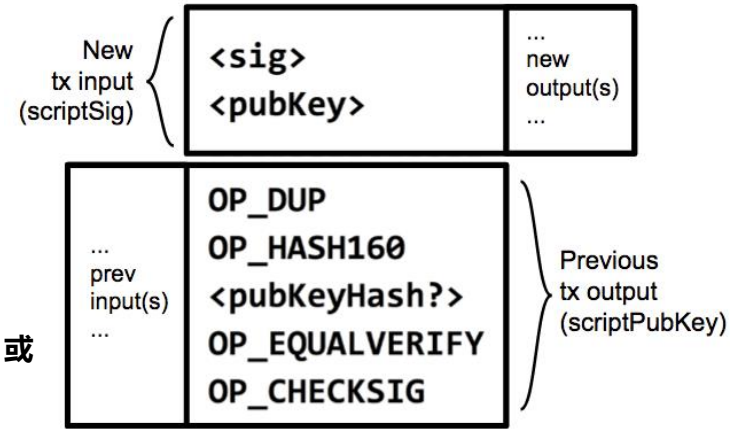
图:
两个事务及其输入和输出脚本



代码执行

1 <sig>
2 <pubKey>
3 op _ dup
4 OP_HASH160
...

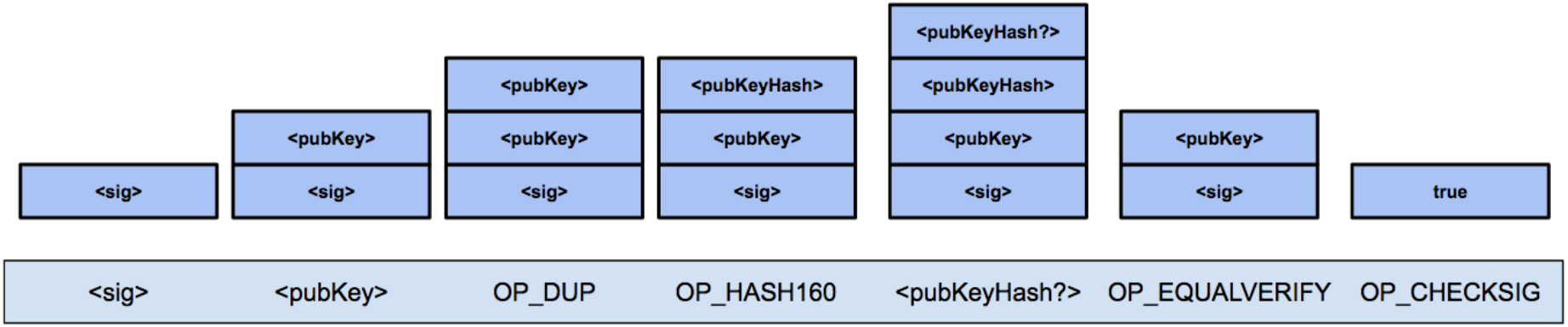
比特币脚本



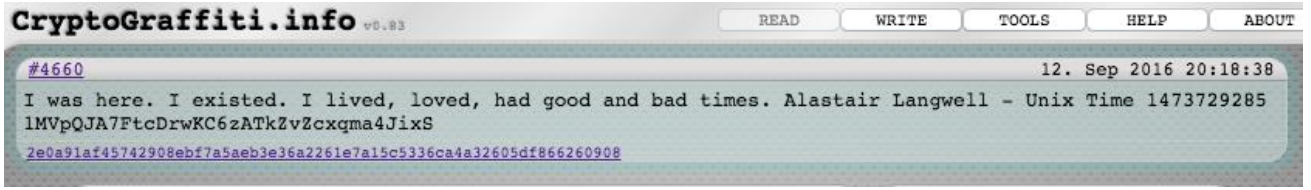
输出显示: "此金额可由
1)**<pubKey>**哈希来解决**<pubkeyhash? >**
2) 加上一个**<sig>**从业主的**<pubKey>**
这将使这个脚本评估到**真**."

专门为比特币构建的名为 "脚本" 或 "比特币脚本语言" 的语言

- **基于堆栈**
- **对密码学的本机支持**
- **简单**



烧伤的证明



如何将任意数据写入比特币区块链?

输出脚本:

OP _ RETURN

< 任意数据 >

<harambe>

烧伤的证明

- 如果到达 op _ return, 则会引发错误
- 输出脚本不能用完, 因此您证明您销毁了一些货币
- op _ return 后的任何内容都不被处理

使用案例

- 证明某事在特定时间点的存在
  - 前。你创造的一个词, 一个文件的哈希/音乐创作的作品
- 引导 calcoin 要求你摧毁一些比特币, 以获得 calcoin

付费-隐藏式哈希与按文本-哈希

在比特币中, 发件人指定脚本。

p2pkkh: 供应商说: "发送您的硬币到哈希这**pubkey**."

- 最简单的情况
- 是迄今为止最常见的情况

但对于复杂的脚本 (如多西格), 这可能是个问题。

供应商说: "要付钱给我, 写一个复杂的输出脚本, 让我使用多个签名。

发信人怎么会知道?

付费-隐藏式哈希与按文本-哈希

解决 方案？**按文本哈希 (p2sh)**

p2pkkh: "发送您的硬币到哈希这**pubkey**."

p2sh: "发送您的硬币到哈希这**脚本**.要兑换这些硬币, 您必须透露具有给定哈希的脚本, 并提供**数据**这将使脚本计算为 true。

- 将复杂的脚本写入文件卸载到收件人
- **p2sh 是比特币自成立以来最重要的改进**

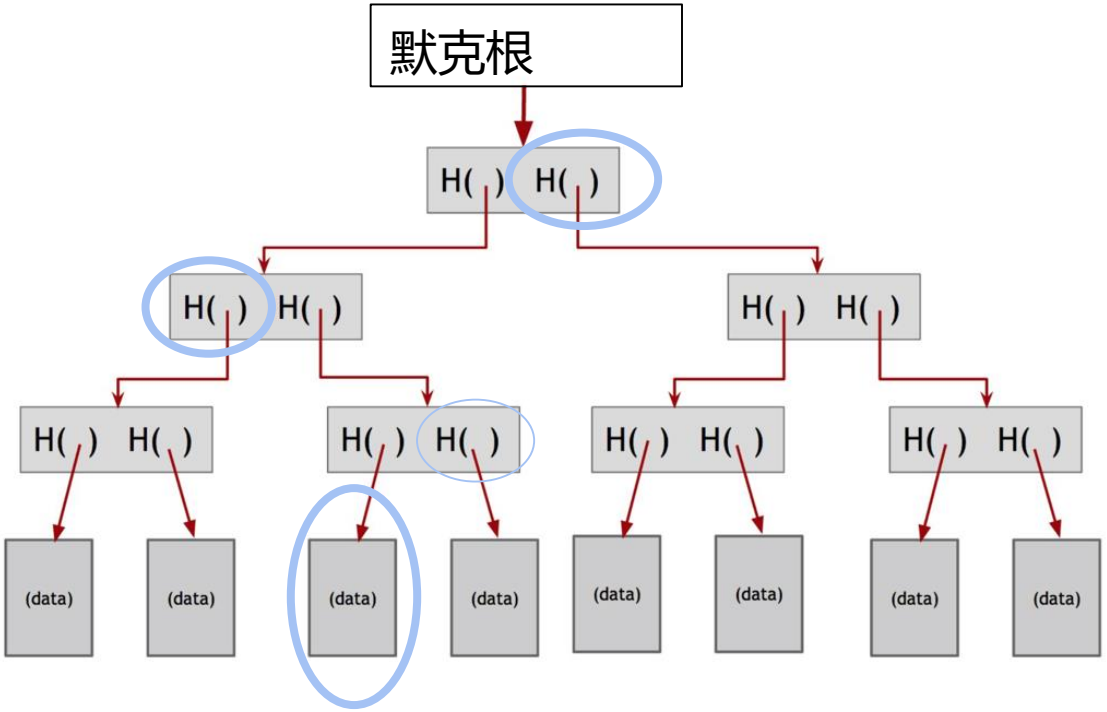
默克树

哈希指针的二叉树

- 哈希被捆绑在一起

这样，您就可以证明您的数据存在于与 merkle 根相对应的树中，而无需实际保存树中的所有数据。

- 例如，如果您在底部的第三个节点中有一条消息，要证明包含在树中，只需将这三个中间哈希保存在树中，并显示您能够使用此数据生成 merkle 根。



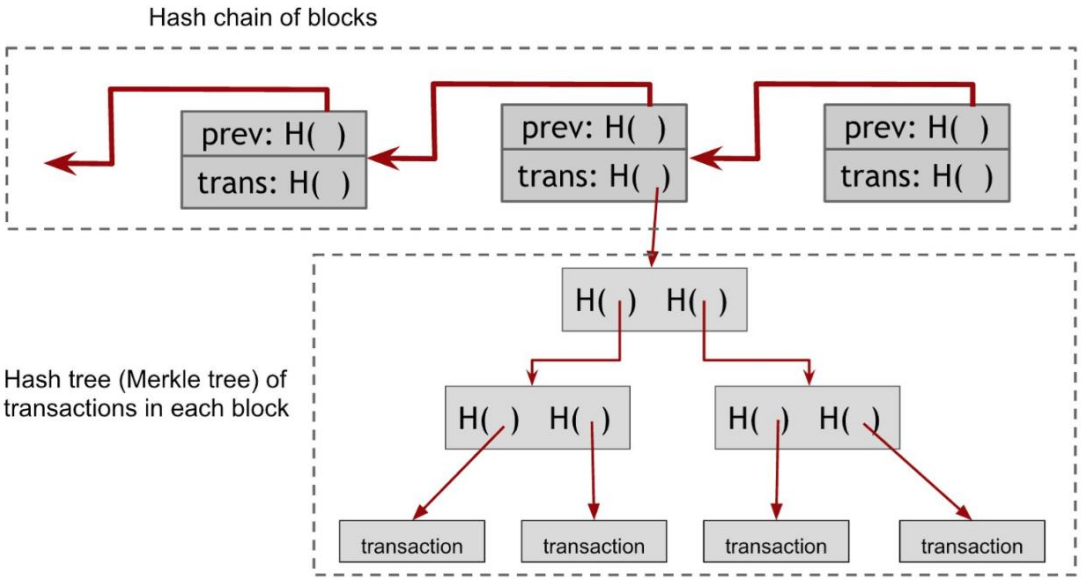
普林斯顿教科书图1。7

默克树-比特币建设

事务是在 merkle 树中的叶子，包括一个硬币基事务

两个哈希结构

1. 块的哈希链
   - 这些方块是相互联系的。
2. 每个方块内部的 txs 的默克树



普林斯顿教科书图3.7/3。8

汞树-采矿，更详细

以前，哈希：

- 默克根
- 防喷
- nonce (不同的价值)

低于一些目标值。

实际上有两个不存在：

1. 在块标头中
2. 在硬币基地

哈希

- 防喷
- 硬币名词 (不同的值)
  - 影响汞根
- 块标头 nonce (变化值)

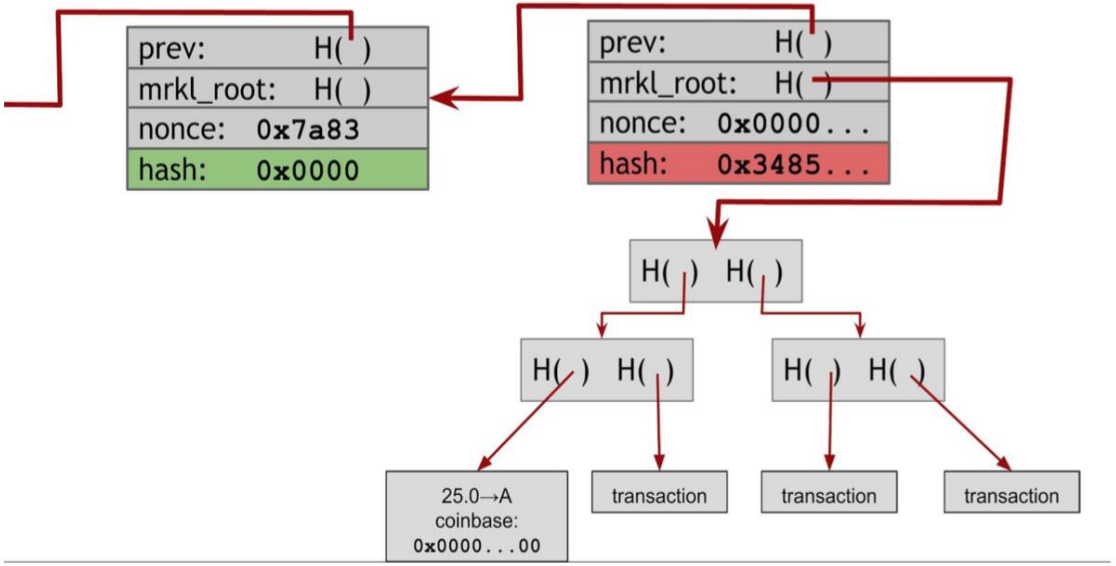


Figure 5.1: Finding a valid block. In this example, the miner tries a nonce of all 0s. It does not produce a valid hash output, so the miner would then proceed to try a different nonce.

普林斯顿教科书

默克树-比特币建设

如果没有解决办法呢?

- 块标头 nonce 为32位
  - ant克斯克斯 sa9 hasths 14 泰铢
  - 尝试所有组合需要多长时间?
  - 2 ^ 32/14 000, 000, 000, 000, 000 = 0.00031秒
  - 消耗 3260次/秒
- 因此, 必须更改默克根
  - 增量硬币开始, 然后再次运行块标头
  - 增加硬币基础的效率降低, 因为它必须传播到树上

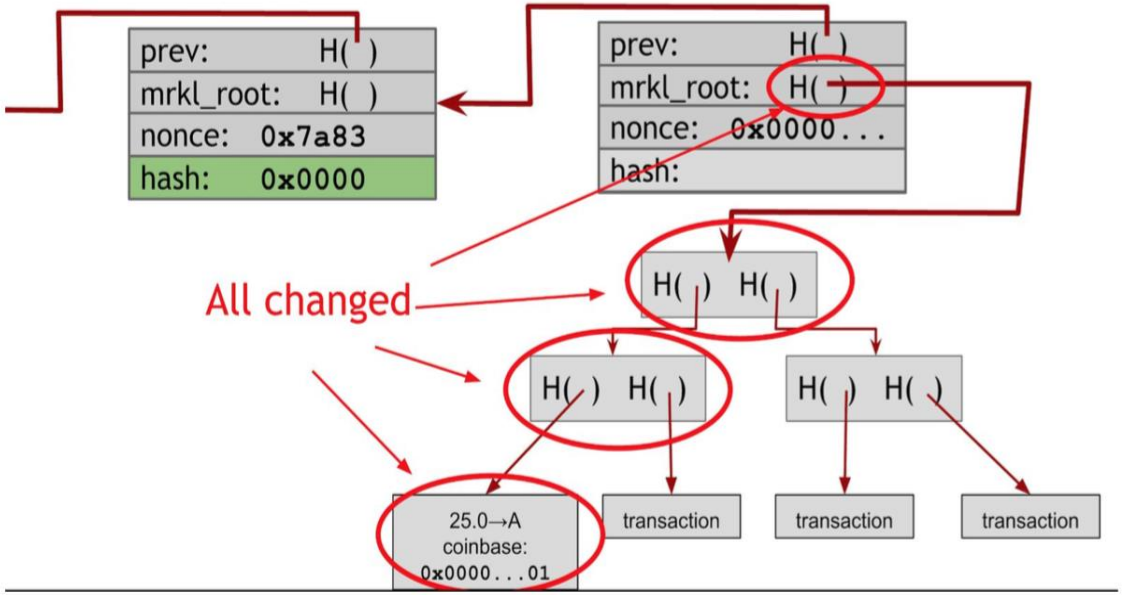


Figure 5.2: Changing a nonce in the coinbase transaction propagates all the way up the Merkle tree.

普林斯顿教科书

简化的付款验证 (spv)

比特币区块链的当前大小: 83.3 千兆字节和不断增长

- 是两年前的300%
- 存储完整的比特币区块链对普通用户来说很可能是不可行的, 那么我们如何解决这个问题呢?

spv-简化的付款验证

- "spv 节点" 或 "瘦" 客户端, 而不是 "完整节点"
  - 不要存储整个区块链
  - 仅存储验证与之相关的事务所需的数据片段
- 网络上几乎所有节点都是 spv 节点

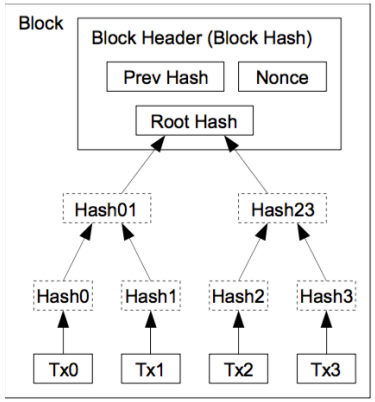
spv-结构

spv 节点仅保留区块链的块头

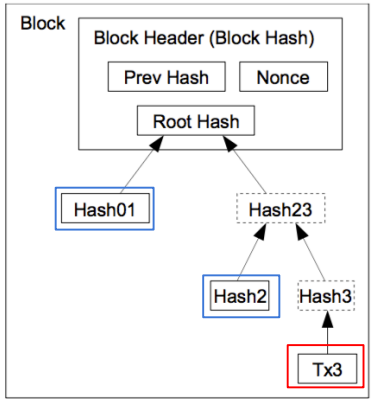
- 通过查询不同的完整节点获得, 直到该节点确信它具有最长的链

如何验证传入 tx?

- 查询完整节点以获取该事务的 merkle 分支
- 哈希 tx 和中间哈希一起获取一个 merkle 根, 请检查它是否与本地保存的根匹配。
- 等待此 tx 有足够的确认, 然后再交付良好的

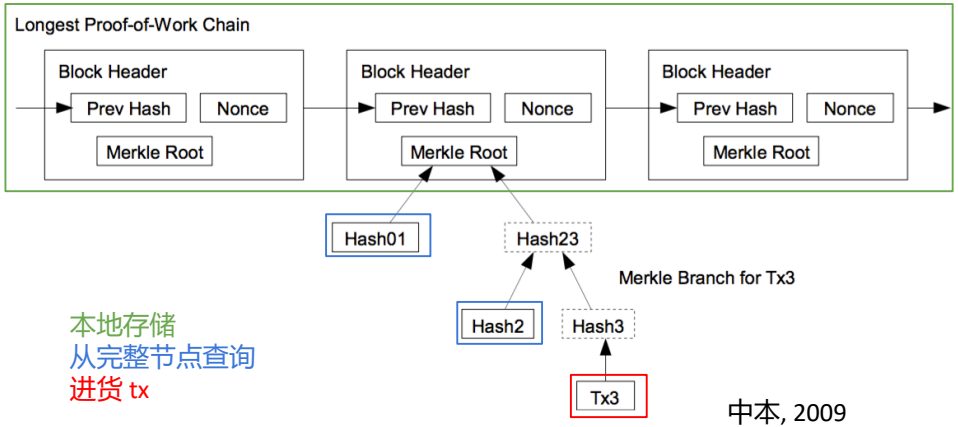


Transactions Hashed in a Merkle Tree



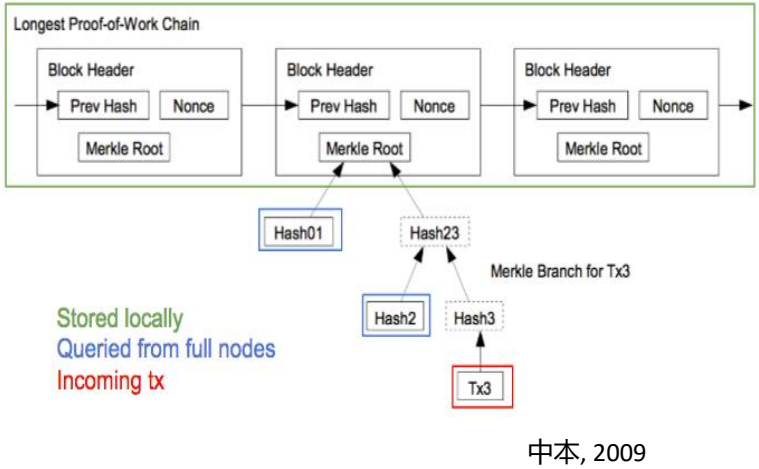
After Pruning Tx0-2 from the Block

中本, 2009



中本, 2009

spv-安全分析



中本, 2009

spv 节点:

- 没有完整的 tx 历史记录, 不知道 utxo 集
- 没有相同级别的完整节点的安全性
- 无法检查块中包含的每个 tx 是否实际上有效

- 只能验证实际影响它们的事务, 并且必须懒洋洋地这样做。

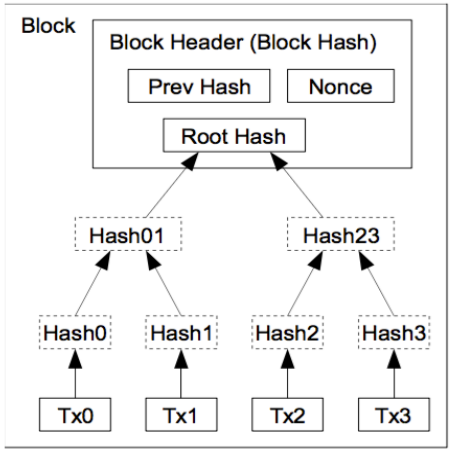
spv 节点假定:

- 传入的块标头不是假链
  - 非常昂贵的攻击 (或任何人) 创建块
  - 长期不可持续
- 有一个完整的节点在那里验证所有事务
  - 这样做有效率方面的好处和激励措施
- ..... 矿工确保他们在街区中包含的交易是有效的
  - 否则, 他们的块将被完全节点拒绝 (非常昂贵的错误!)

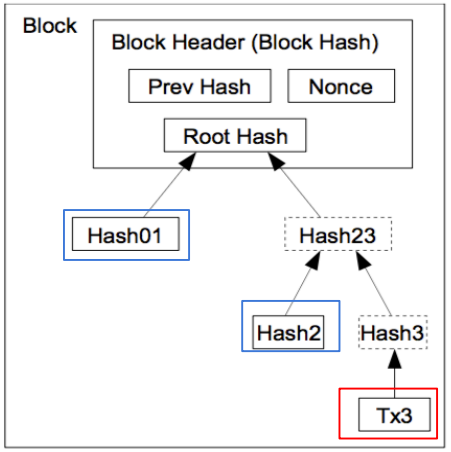
spv-节省成本

极大的成本节约

- 块头只有 ~ 一半的大小
  整个区块链
  - 83 mb 与83兆字节



Transactions Hashed in a Merkle Tree



After Pruning Tx0-2 from the Block

中本, 2009

spv 节点利用获取数据懒洋洋地验证事务

- 对于必须验证许多 txs 的大商人来说, 这可能是一个问题

对于比特币的大多数消费者和用户来说, spv 是一个不错的权衡。

धन्यवाद

Hindi 印地语

Спасибо

俄语

شكراً

阿拉伯语

格拉齐

意大利语

நன்றி

Tamil

泰米尔语

以任何努力

多谢

繁体中文

谢谢

英语

多谢

简体中文

ありがとうございました

日语

ขอบคุณ

泰语

谢谢

西班牙语

奥布里加多

西葡萄牙语

丹克

德语

谢谢

法语

감사합니다

朝鲜语