

第一章

暗号理论和假想货币入门

1月10日

通货

- ① 有控制供给的方法
- ② 防止不正当行为的安全网：法律的规制

假想货币

- ① 防止系统的状态被串改 システムの状態の改竄を防ぎ
- ② 前后矛盾，对于不同的对象前后不一致的主张 二枚舌（にまいじた）
比如不能用统一货币支付两个人

与法定货币（Fiat currency）不同的是，假想货币并不依赖于中央集权的权威性，纯靠技术实现

假想货币也成为密码货币

1.1 密码学中的哈希函数 cryptographic hash function

属性

- 1. 对于任意大小的输入文字序列都能使用
- 2. 生成已决定大小的文字序列
- 3. 能进行高效计算 $O(n)$
- 4. 耐冲突性（collision）
- 5. 匿名性（hiding）

6. 难题亲和性 (puzzle friendliness)

1.1.1 属性1：耐冲突性

- 即两个不同的输入不会得出能被发现的相同输出
- 冲突是不可避免的，需要做到的是要使冲突不被找见

如何寻找冲突

birthday paradox：可能输出个数的平方根个数作为输入验证，可以找到冲突

对于256bit的数据来说，计算出全部的哈希值是要消耗大量时间的

但是也有对于特定的哈希函数高效的找到冲突的办法，如：

$H(x) = x \bmod 2^{256}$ 这个函数只输出输入值得最后256bit

应用：message digital

用户将文件上传，下载的时候保证文件的安全性与完整性

Alice：记住原本文件的哈希值，在下载的时候，计算下载文件的哈希值，并与之比较

1.1.2 属性2：秘匿性

已知输出，不能得到输入

抛硬币游戏 (コイントス)：抛出正面="heads" 抛出反面="tails" (=为发表哈希值)

要实现隐秘性，x的正解候补值不能存在。也就是说，x必须在某种极大分散的元素的集合中选择

如果必须从元素不分散的集合中选取，就要连结其他散列元素集合 (|| 是连结符号)

「秘匿性」(hiding): 「min エントロピー」(min-entropy)の高い確率分布から秘密の値rが選ばれているときに、 $H(r||x)$ が与えられても、xを実現的に見つけることはできない。

「min エントロピー」最小熵 结果能预测到多少

最小熵大，分布分散

应用：commitment scheme 承诺系统

[承诺系统讲解](#)

「コミットメントのスキーム」 commitment scheme

- $com := \text{commit}(msg, nonce)$

commit函数，输入是 message和被称作nonce的随机取值，作为commitment返回

nonce 为只能使用一次的值，且为随机的256bit值

- $\text{verify}(com, msg, nonce)$

verify函数，输入是com，msg，nonce

如果 $com = \text{commit}(msg, nonce)$ 为真，反之为假

属性：

- 「秘匿性」：即使给了com也无法实际找到msg
- 「拘束性」： $msg \neq msg'$ 且 $\text{commit}(msg, nonce) = \text{commit}(msg', nonce')$ 这样的对找不到

怎么证明具有如上两个属性呢？

$\text{commit}(msg, nonce) := H(msg || nonce)$

值得注意的是，即使满足拘束性基础的哈希函数也不一定满足耐冲突性

1.1.3 属性3：パズル親和性 谜题友好

「パズル親和性」(puzzle friendless)：k在最小熵很大的概率分布中选择的时候，哈希函数H的全部n bit的输出y，比 2^n 更快的，在极端时间内能找到满足 $H(x||k)=y$ 的x是不可能的的时候，称为哈希函数H的难题亲和性

简而言之，即使谁想从哈希函数中获取特定的输出值y，如果输入的一部分是合适程度的随机选择的话，能得到目标y的输入x是极难找到的

应用：探索puzzle

在非常大的空间中寻找解的数学问题中没有有效求解办法

「探索パズル」(search puzzle)

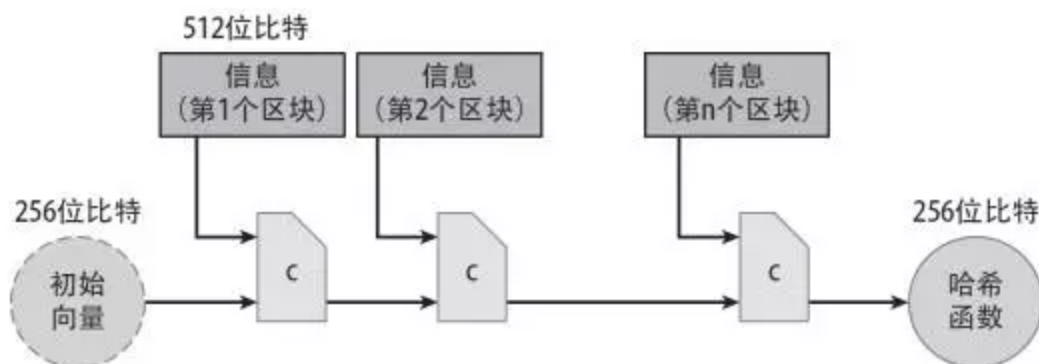
- 哈希函数 H
- id：从最小熵极高的集合中选择的值
- target集合 (目标集合) Y

$$H(id \parallel x) \in Y$$

1.1.4 SHHA-256

「Merkle-Damgard 变换」：能将任意长度的输入进行哈希函数计算

「压缩関数」：对于固定长度的输入有耐冲突性的哈希函数



比如压缩函数代入长度为m的输入值，并产生长度短一些为n的输出值。哈希函数的输入 (可为任意大小) 被分为长度为m-n的区块。MD变换运作过程如下：将每个区块与之前

区块的输出一起代入压缩函数，注意，输入长度则变为 $(m-n)+n=m$ ，也刚好就是压缩函数的输入长度。对于第一个区块而言，之前没有的区块，我们需要选取一个初始向量（见图1.3）。每次调用哈希函数，这个数字都会被再一次使用，而在实践中，你可以直接在标准文档中找到它。最后一个区块的输出也就是你返回的结果。

[区块链：技术驱动金融（十）特性3——谜题友好](#)（与书上内容完全一致）

哈希函数建模

为了保证安全性，不同的应用会要求不同的哈希函数特性。要确定全面可证明的一系列哈希函数特性是十分困难的。

SHA-256是接受768bit的数据，用压缩函数输出256bit

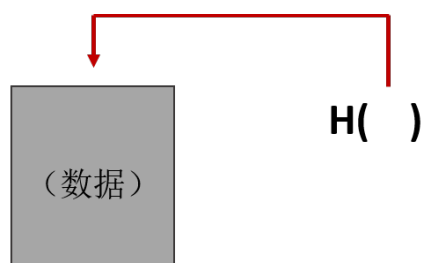
1.2 ハッシュポインタとデータ構造

[哈希指针与哈希结构](#)

哈希指针是一个指向存储地点的指针，加上一个针对存储时信息的哈希值。

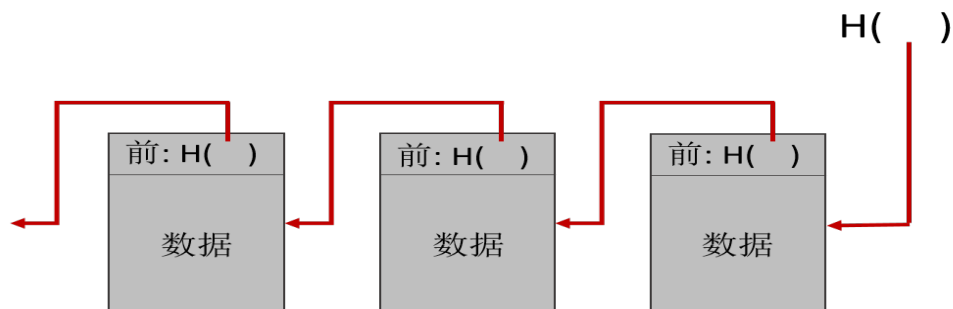
哈希指针不仅是一种检索信息的方法，同时它也是一种检查信息是否被修改过的方法。

可以用哈希值构造何种数据结构，核心思想是用哈希指针代替普通指针



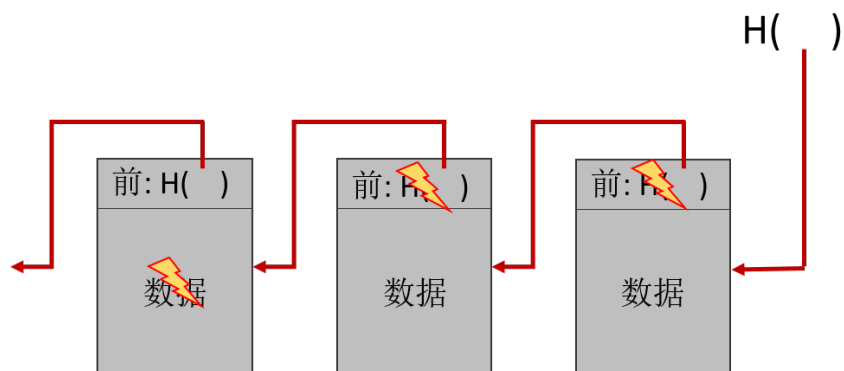
1.2.1 ブロックチェーン 区块链

用哈希指针构造的链表



我们存储列表的开头，这是一个指向最近数据块的常规哈希指针。

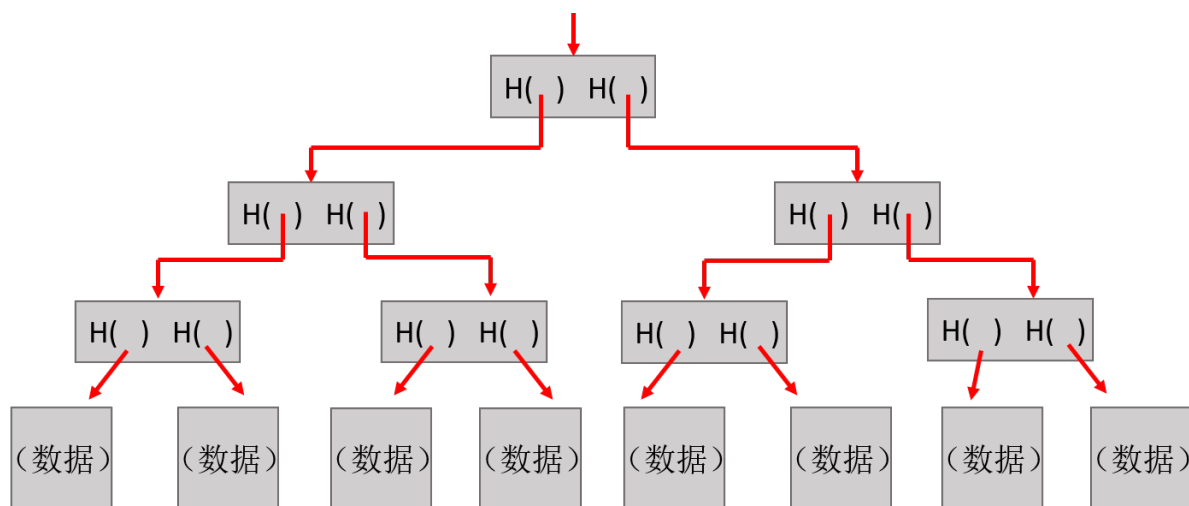
可以通过哈希值来检验前一个值是否被篡改



如果敌人想篡改k块的内容，就会引起与k+1块的哈希值不一致。可以一直修改之后的哈希值，但是最终还是会遇到我们储存的哈希链表的头部。因此仅仅通过记住单独这个哈希指针，我们就记住了整列防篡改哈希。

- 链表开头的块称为 **创世块「ジェネシスブロック」(genesis block)**
- 这与Merkle-Damgard变换十分类似，实际上两者在安全性上也有类似的部分

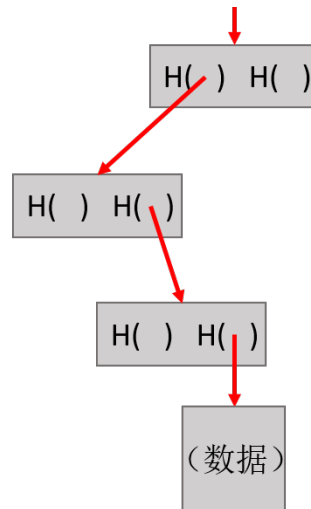
1.2.2 Merkle树 (Merkle木)



只需保存根节点的哈希指针

哈希指针与哈希结构

1.2.3 成员证明 「メンバーであることの証明」 (proof of membership)



一个树上有 n 个结点，只需要展示 $\log n$ 个块 所以时间会很快

「ソート済みMerkle木」 排序Merkle树

底层数据以一定方式排序的Merkle树

1.2.4 非成员证明

利用排序Merkle树，我们可以在对数大小的时间空间里证明某个块不属于Merkle树

展示该块前一个块和后一个块的哈希路径，如果这两个块在树里是连贯的，那么这就表明了我们想要证明的块不属于Merkle树。

哈希指针可用于任何无环有指针结构

1.3 电子署名

电子署名（翻译版）

属性：

1. 只能自己签名，其他任何人都可以确认其是否有效
2. 署名与特定的文书连接，其他文书不能使用来证明你是否同意

数字签名体系：一个数字签名体系包含以下三个算法

- $(sk, pk) := \text{generateKeys}(\text{keysize})$

`generateKeys`方法是，输入密钥大小，生成密钥对。

私钥`sk`，因为用于信息署名，所以非公开管理

公钥`pk`，发送给全部相关人员，用于确认

- $\text{sig} := \text{sign}(sk, \text{message})$

`sign`方法，输入信息，获取私钥`sk`，对`message`署名

- $\text{isValid} := \text{verify}(pk, \text{message}, \text{sig})$

`verify`方法，输入`message`，签名，公钥，返回理论值`isValid`

属性

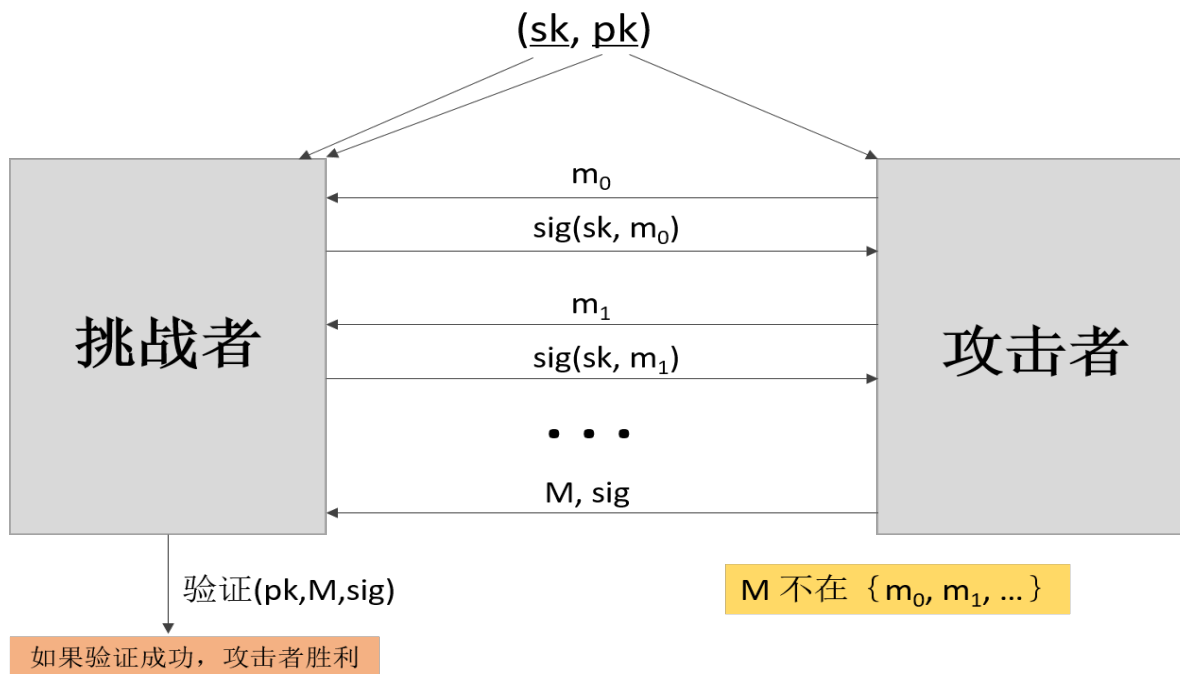
1. 正确的签名要能通过验证

$\text{verify}(pk, \text{message}, \text{sign}(sk, \text{message})) = \text{true}$

2. 签名是存在且不可伪造的

`generateKeys`和`sign`使用随机幻术，验证过程是确定的

不可伪造游戏



挑战者对此宣称进行验证
使用generateKeys生成密钥

sk, pk

攻击者声称能伪造签名

pk

允许攻击者能够获得他选择文件上的有效签名

名

只要他想，在合理的推测范围内，就认为他可获得多个文件的签名

合理推测范围：100万次推测

用渐进法的说法，就是认为密钥大小的多项式函数的推测数，不承认在此之上的

攻击者判断已经足够后，选择未署名的明文，自己署名后，发送给挑战者

挑战者验证是否有效，如果有效则
证明攻击成功

1.3.1 实际问题

电子署名（翻译版）

1. 随机函数
2. 对信息的哈希值进行签名
3. 可以对区块链的开头哈希指针进行签名，不仅保护了哈希指针本身，对整个区块链都进行了保护

1.3.2 ECDSA

椭圆曲线数字签名算法「橢圓曲線DSA」(Elliptic Curve Digital Signature Algorithm)

9.9.1 椭圆曲线：

三次多项式 $f(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ 没有重根的时候

$$E : y^2 = f(x)$$

成为椭圆曲线，当上式中的 x ， y 都是有理数时，称为点 (x,y) 为有理点

其是无奇点的；亦即，其图形没有 尖点 或 自相交。

9.9.2 有限的椭圆曲线

p 为 5 以上的素数

[椭圆曲线群构造](#)

有限体 $GF(p)$ 上的椭圆曲线式 E 有如下标准型

$$E: y^2 \equiv x^3 + ax + b \pmod{p} \quad \text{当 } 4a^3 + 27b^2 \neq 0$$

有理点加上无限远点构成集合 $E(p)$ ， $E(p)$ 的要素书成为 $E(p)$ 的位数 $E(p)$

$$p + 1 - 2\sqrt{p} \leq E(p) \leq p + 1 + 2\sqrt{p}$$

比特币使用基于标准椭圆曲线“secp256k1”的ECDSA，“secp256k1”预计能够提供128位的密保。

ESDSA中使用很好的随机函数是十分重要的。即使只是在签名的时候随机性不好（生成密钥时的随机性），密钥完整的时候，也会根据签名泄露密钥（这是根据DSA的普遍缺点，并不是ESDSA单独的）

1.4 作为信息识别的公钥

[公钥作为身份](#)

认为pk是一个身份，当相声称一个新的身份时，只需通过generateKeys来生成私钥sk和公钥pk，公钥pk对于其他人来说就是你的身份。实际上，公钥pk很长，使用pk的哈希值来识别。当你发送信息，别人来确认是你发送的信息时，需要一下两步

1. 检查pk的哈希值是否符合你身份
2. 检查信息在公钥验证下是有效的

1.4.1 去中心化身份管理

不需要去由中央集权存在的中心注册，自己可以完成注册，不需要向别人声明已注册的名字，也不需要让别人知道自己使用特定的名字

可以生成任意多个身份，想要保证自己的隐秘性，可以生成一个新的身份，并在一段时间后扔掉

这种身份认真信息被称为adress 地址，是公钥的哈希值

不同的人生成两个相同的身份是十分困难的

必须使用高性能的随机数生成器

虽然费中央集权中心化，能保证匿名性，但是还是可以通过身份的一系列的动作来判断现实身份

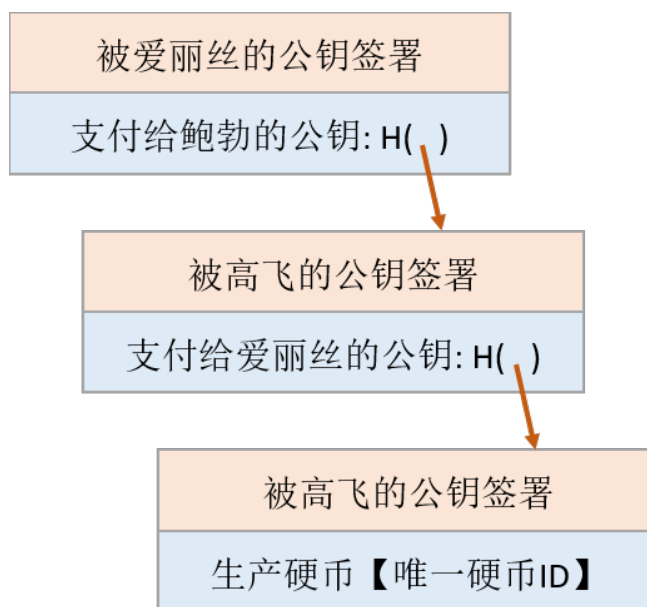
1.5 两个简单的假想货币

[两个简单的加密货币](#)

1.5.1 Goofycoin 高飞币

高飞币的规则：

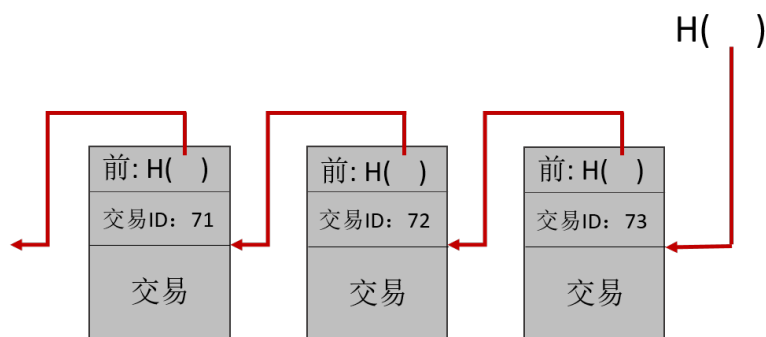
1. 高飞可以通过简单地签署一个声明他创造了一个带有独特ID的新币来创造新币。
2. 任何拥有硬币的人都可以通过签署一个声明将其转移给其他人：“把这个硬币转移给X”
(在这里X特指公钥)
3. 任何人都能验证硬币的有效性，只要随着哈希指针链回到高飞开始创造币的地方，沿途验证所有的签名即可。



安全问题：

双花攻击(double-spending attack) 「二重払い攻撃」

1.5.2 Scroogecion 史高治币



每个块都有一个交易的ID，交易的内容，以及一个指向前一个块的哈希指针。史高治会持有持有代表整个结构的哈希指针（最后一个哈希指针），然后他将签署并公布这个区块链。

在史高治币中，一个交易只能当其在史高治签署过的区块链中时才能生效。任何人都能够检验交易是否是史高治批准的，只要验证交易出现的块里史高治的签名即可。史高治确保他不会签署一笔试图双花的交易。

两种交易

1. 创造硬币交易

交易ID: 73			类型: 创造硬币
硬币创造			
编号	价值	接受人	
0	3.2	0x	← 硬币ID 73(0)
1	1.4	0x	← 硬币ID 73(1)
2	7.1	0x	← 硬币ID 73(2)
.....			

创造硬币交易。这个创造硬币交易创造了很多的硬币，每个硬币在交易里都有个连续的编号。每个硬币也有一个价值，它值一定数量的史高治币。最后，每个币都有一个接受者，是硬币被创造出来时得到硬币的公钥。所以创造硬币创造了一连串不同价值的新硬币，然后作为最初的拥有者签署给其他的人。硬币都有硬币ID，一个硬币ID是一个交易的组合ID和交易里硬币连续的编号。

2. 支付硬币交易

这个将消耗一些硬币并摧毁它们，然后创造出相同价值的新币。新币可以属于别人（公钥）。这个交易必须由支付的人进行签名，所以如果你是硬币的拥有者并且想在这笔交易中花掉它，那么你需要数字签名这个交易来说明你同意花掉这个硬币。

支付硬币交易在下面四个事情真实的情况下是有效的：

1. 被消费的币是合法的，即它们是上一个交易合法制造出来的
2. 被消费的币没有在之前的交易中被支付过，即没有双花支付
3. 这个交易支出的消费总额等于之个交易产生的新币总额
4. 具备所有被消费币的所有者的合法签名

条件满足，交易有效---->Scrooge将交易加入区块链---->之后任何人都能知道交易已发生---->参加者接受交易已经发生

被发表前，它都有可能被一个双花交易所取代，即使它已经被前三个确认为有效了。

取引 ID : 7 3 タイプ : PayCoins		
消費されたcoinID : 68 (1) , 42 (0) , 72 (3)		
作成されたコイン		
番号	価値	受領者
0	3.2	0X...
1	1.4	0X...
2	7.1	0X...
署名		

这个系统中货币不变，然而可以通过消耗旧币，生成新币来分割，结合货币

但是Scroogecoin是中心化的货币

第二章 bitcoin非中心化结构

2.1 中心化和vs中心化

中心化和vs中心化

去中心化并不总是非此即彼的。几乎不存在这样的系统，是纯粹中心化或者纯粹去中心化的。

bitcoin和别的货币兑换的时候，存储在自己的钱包的时候，各个软件也是各种程度上的中心化和去中心化的结合。

bitcoin去中心化的五个问题

1. 谁管理交易账本
2. 谁有权利判断交易是否有效
3. 谁能制造新的bitcoin
4. 谁能决定系统规则的更改
5. bitcoin如何获得交易价值

bitcoin在不同的实现上有着不同的中心化/非中心化分布：

1. p2p网络接近纯粹的去中心化
2. 挖矿：是高度中心化，因为消耗资本高，大多数由比特币社区执行
3. 比特币节点的软件升级。

这关系到系统的游戏规则在何时做出怎样的改变。可以想象，有许多相互操作的协议，比如电子邮件。但在实践中，大多数节点用于参考，开发者受到社区的信任，并给予其极大的权力。

2.2 分布式共识 分散型合意形成 distributed consensus

分布式共识协议。有n个节点，每个节点都有一个输出值。这些节点中有一些是错误的或是恶意的。一个分布式协议有下面两个属性：

- 以每个正确的节点都得到相同的值为结束
- 这个值必须是一个正确节点产生的

p2p网络中alice将交易信息向网络中所有节点进行广播，bob是否在节点中运行并不重要

各个节点要达成共识的是：

1. 被广播的交易是什么
2. 交易以什么顺序进行

系统生成一个全局的账本（以区块链的块为单位达成共识）

- 在任何时刻，在p2p网络中各个节点都有一个包含一个由已达成共识的交易区块的序列的账本
- 另外，每个节点还包含一个已经收听到但是还没包含在区块链中的交易集合
- 因为网络的不完善，会有一个节点收到还未被其他节点收到的消息

共识达成解决办法：

- 以一定的间隔，系统内的全部结点，提出自己未完成的交易集合成下一个区块
- 然后各个节点执行一些共识达成协议，在协议中每个输入都是自己出的区块
- 即使有效区块只被一个结点提出，则认为是合法输出
- 违背这个区块包含的未完成交易可以进入下一个区块

对于bitcoin这个方法的问题：

1. 有恶意的结点，达成共识困难
2. 网络不完全：
 - a. 延迟：

延迟高的话，没有全球时间的概念全部的结点无法根据简单基于时间戳判断交易发生的顺序。因此共识协议不能包含这样形式的指令：“在第一步中发送第一条信息的节点

在第二步中怎样怎样”，这是行不通的，因为不是所有的节点会同意在协议的第一步，哪一条信息是首先被发送的。

b. 节点崩溃

2.2.1 不可能性

「ビザンチン将軍問題」(Byzantine Generals Problem)：有1/3的将军背叛，就不能达成共识

「FLP不可能性」：在某些特定的情况下，即使只有一个缺陷，也无法达成共识

2.2.2 以前的前提条件

因为bitcoin没有之前论述的模型，所以意外的能较好的达成共识

- 引入激励模式：并没有从一般性上解决分散性共识达成问题
- 包含随机性的概念：没有特定的开始时间和结束时间，而是依据一段很长的时间来达成共识，随着时间推移共识达成的越来越好

2.3 使用区块链的不明身份的人的共识达成

bitcoin的结点 没有可持久化，长期的身份标识

为什么没有身份标识的原因：

1. 在p2p系统中，即使给参与者身份标识，也没有中心权威来保证参与者不会随机生成新的结点
 - 「結託攻撃」(sybil attack)：任意创建新的节点的攻击，一个攻击者伪装成多个参与者进行操作
2. 使用假名是bitcoin的固有目标
 - 虽然会有一个人进行的多个交易会被联系在一起，并不是说能保证极强的匿名性，但是可以肯定的是不强制暴露参与者的现实的真实身份呢

如果有身份标识的话，设计会变得更简单：

1. 因为没有身份标识，协议能使用的命令变少
 - 例如：不能使用类似【现在拥有id最小的结点执行某些步骤】的命令
2. 安全属性：被赋予身份标识的结点不能被赋予新创建的结点的身份，安全问题变得简单

因为这些原因，bitcoin的一致性协议变得困难

Weaker assumption：随机选择结点

有这样的能力，无论用什么方法，都可以在系统中挑选一个随机结点

令牌 token トークン

2.3.1 隐式一致性

在每一回合，都能随机选择结点，这个节点提出链的下一个区块

回合：区块链中的不同区块

别的结点，自动判断是接受，还是拒绝这个区块。如果接受，则将这个区块加入区块链来扩展。

若果拒绝，则无视这个区块，从随后一个加入区块链的区块开始扩展

[bitcoin的整体一致性算法「ビットコイン同意形成アルゴリズム」\(单纯化版\)](#)

[Consensus algorithm \(simplified\)](#) 以随机选择结点为前提的简化

1. 向所有节点广播新生成的交易
2. 每一个节点都收集一个还没被加入到区块链的没完成交易链表
3. 在每一轮中，随机选择一个节点广播提议下一个区块 (将监视到没完成交易的集合)
4. 其他节点，表示是否接受这个区块 (非双花攻击，有有效署名)
5. 通过下一个区块包含该区块的哈希值表示接受，或者包含前一个区块认为的合法哈希值

攻击

bitcoin的盗取

- 不能成功，因为不能伪造他人的签名

Dos攻击

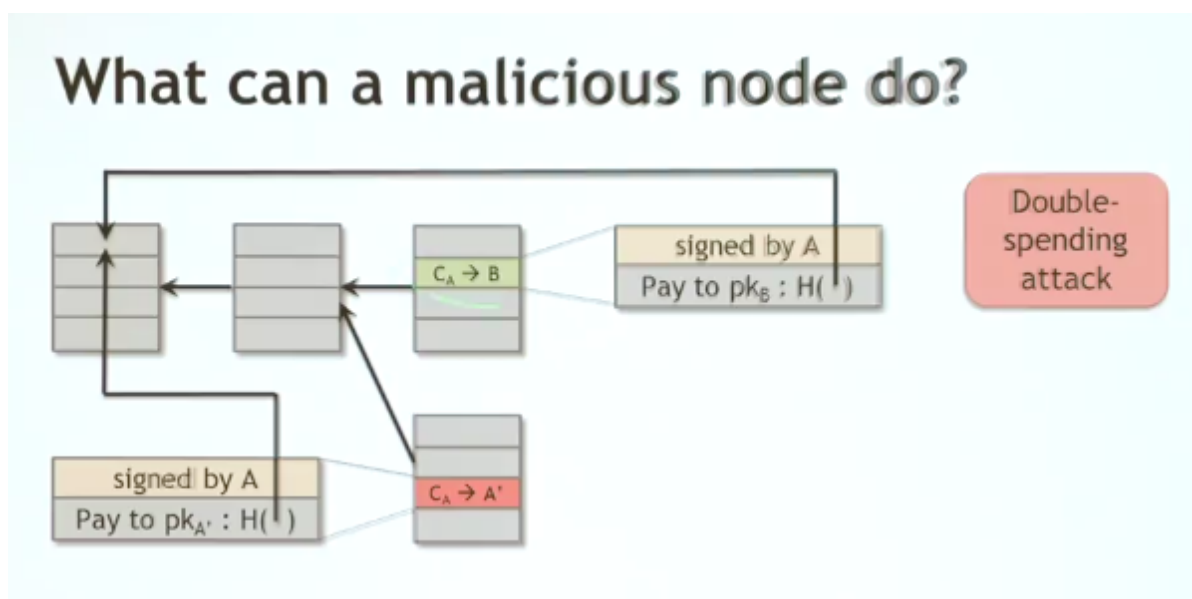
唯一合法的做法，alice拒绝bob的协议，bob只需等到下一个区块，总会有诚实的结点将区块加入区块链中

二重消费攻击(double spending attack) 「二重Dos攻撃」

交易中Alice的署名，向bob的公钥支付的说明，包含哈希值的数字构造

在这之中，至少有两个以上的哈希值：

1. 指向前一个区块的哈希指针
2. 使用的coin是从之前哪个交易中获取的哈希指针
3. 实际上，还有一个Merkle追溯



二重dos攻击是否成功取决于alice->bob->alice或者alice->alice->alice谁最终进入了长期一致性链，健全的结点按照更长的有效分支延伸

在技术并没法判断哪个是合法的，很有可能二重攻击生效，即红色的区块进入了长期一致性链中，下一个结点的哈希指针是红色区块的哈希值

通常情况下，选择最先检测的区块进入长期一致性链，但因为延迟的原因，很可能第二个提交的交易先被检测

被忽视的区块被称为孤儿区块 (orphan block) 「孤独ブロック」

bob如何保护自己不被二次消费攻击

「ゼロ承認取引」 零确认交易 (zero confirmation transaction) :

一直在网络上监听，已收到交易信息就立刻完成精算，允许alice下载软件

或者可以等到交易在区块链上获得了一个认证为止，意味着至少有一个节点创建了区块，并提议了该区块进入区块链

如前面所述，即使第一个区块被确认，Alice仍可能在之后再广播一条交易，进行二重消费攻击

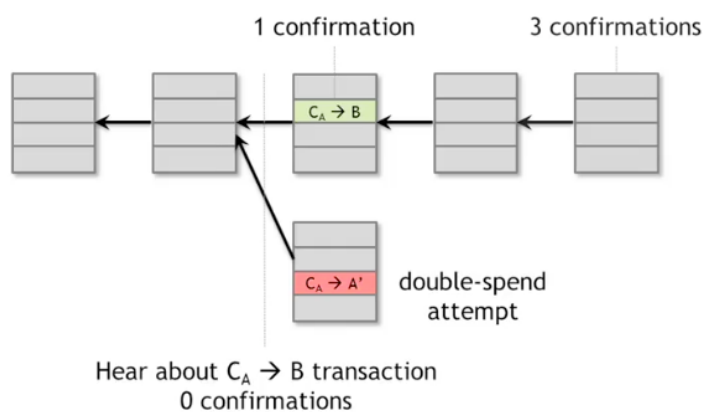
保护方法

当bob意识到alice->bob的交易变成了孤儿区块，就放弃交易

或者，当发生二重消费攻击，不立刻允许Alice下载文件

当bob关心的区块被下一个区块所扩展，就是第二个确认

From Bob the merchant's point of view



二重消费交易的剩存的概率，随着确认数成指数级下降

根据经验，一般等待6个确认：

- 必须等待的时间

- 能进入一致性连的必要时间

2.4 インセンティブとPOW (Incentives and proof of work) 工作量证明

bitcoin的去中心化，一部分以技术实现，一部分依靠于巧妙地激励措施

用激励措施鼓励这些正常的结点

在二重消费攻击中，我们在技术上无法分辨哪个是真正正确的。同时我们，也不知道节点的身份信息。这使得我们处罚那些恶意结点变得十分困难。

与之相反，我们给予那些已经达成长期共识在区块链中存在的结点一些报酬，因为不知道节点的具体身份，用能使用的电子货币来取代，即向其支付一定数量的bitcoin。

用这种分散式共识达成过程的构建应用就是货币。具体说就是，给予根据支付货币采取正确行动的结点一定的奖励。

2.4.1 制作区块的报酬

bitcoin有两种独立的激励机制

Incentive 1: block reward 区块奖励

在bitcoin的规则中，生成区块的结点会包含一个特殊的创币交易，交易的地址一般选为自己的地址，即将自己创造的货币交易给自己。作为将创币区块加入至共识链中的交换，或定一定报酬。

伪代码

block(t) =

{

前一个区块的散列值，即hash(block(t-1))

区块所包含的交易信息 第一笔交易只有收款人地址，

区块奖励金额 任意随机数：随机数满足 $\text{hash}(\text{block}(t)) < \text{特定值}$

}

bitcoin最初报酬50货币，每生成21万个区块，报酬减一半，即四年一减半。

只有创造能进入到长期共识链中的区块的节点才能获得报酬。bitcoin总额2100万bitcoin。挖掘bitcoin报酬到2140会结束。

2.4.2 交易手续费

incentive 2 : transaction fees

[精读比特币白皮书（激励部分）](#)

[比特币技术笔记（4）区块链&公共账簿](#)

除了区块链激励外，矿工还可以获得手续费收入。在一个『区块』所记录的交易中，收款金额可能小于付款金额：差额即交易手续费。当然，比特币的用户可以选择不支付手续费，但没有手续费的交易很可能会被矿工抛弃而不被记录到任何『区块』中，成为无效交易。因此在实践中，大多数人都会支付一个非常低的手续费，比如0.001比特币。

因为区块激励会变少，所以交易费就变得越来越重要。但是这个激励系统会如何演变，依赖与很多游戏理，很多还没有完全解决。现在是一个有趣的开放[研究领域](#)。

2.4.3 挖掘和工作量证明

共识达成机制的问题：

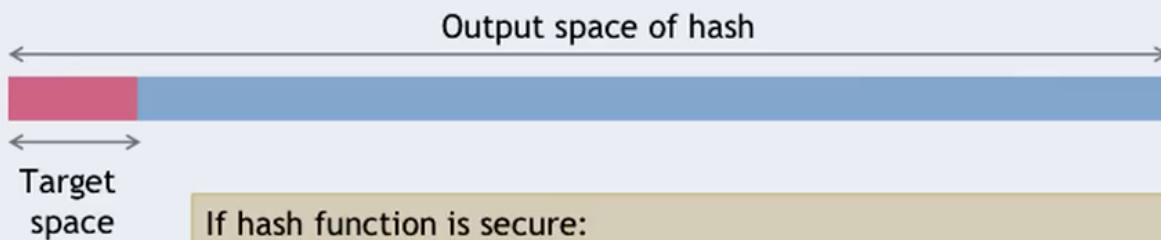
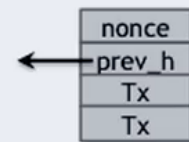
1. 怎么样任意选择节点
2. 怎么解决由激励机制产生的混乱，因为所有节点都想获得报酬，系统又不安定的危险性
3. 解决欺骗性问题。为了破坏共识达成过程而产生大量的合规节点

pow：以谁都不能独占的资源的比例来选择节点，近似于随机选择。基于算力的Pow系统，基于所拥有货币比例的proof of Stack (权益证明) 系统。

```
block(t)
{
    nonce
    message
    prev_hash(block(t-1))
}
```

Hash puzzles

To create block, find nonce s.t.
 $H(\text{nonce} \parallel \text{prev_hash} \parallel \text{tx} \parallel \dots \parallel \text{tx})$ is very small



If hash function is secure:
only way to succeed is to try enough nonces until you get lucky

block(t)所计算出的hash值必须要满足一个很小的值，即要在比hash函数全部输出空间小很多的目标空间中。必须满足以下不等式。

$$H(\text{nonce} \parallel \text{prev hash} \parallel \text{tx} \parallel \text{tx} \parallel \dots \parallel \text{tx}) < \text{target}$$

nonce的作用是将找到满足条件的哈希值这一过程变得更加困难一些，如之前说的哈希函数的难题亲和性，想要找到一个合适的输出值，必须一个一个试nonce值。

所以如何随机选择结点就变成了，大家独立解决难题，找到合适的hash值，当找到了一个合适的nonce之后，这个节点就可以提出区块。

2.4.4 计算困难

计算难题，要满足的三个性质

三个属性

1. 计算必须极为困难：挖掘资本很高，所以形成算力集中。
2. 成本参数化
3. 其他节点能简单认证pow计算正确

2.4.5 成本参数化

bitcoin p2p网络中所有节点每创造2016个区块计算一次目标值。

$$\text{目标值} = \frac{\text{目标空间大小}}{\text{输出空间大小}}$$

bitcoin中大约每隔十分钟产生一个区块，即大概每两周计算一次目标值。

一个节点获得下一个区块的概率=她所拥有的哈希算力在全网中的比值。

10分钟并没有特定的意义。间隔时间过短将导致，效率低下，没办法将多个交易加入一区块中。

关于矿工造作的两个模型：

在分散系统和计算机安全领域，假设一部分节点是健康的，即使其他节点任意操作系统也能正常运行。虽然这里我们采取的方法看起来是基本的，但是在计算大多数的时候以哈希算力给节点所加的权重是不同的。

但是，在游戏理论中，为了判断系统采取了什么操作，采取了完全不同的实现度更高的方法。不分别节点是否健康。假设所有节点都遵循自己的激励模式操作，每个节点考虑其他节点的战略，而随机选择是自己利益最大化的战略。如果激励系统和协议涉及良好，所有的节点在任何时候都会遵循规则。也就是说，健康的操作只是所有战略中的一种而已。并没有道德上的对错。

游戏理论中最大的问题是，默认矿工的操作是否是纳什均衡点，也就是是不是有采取恶意形动的节点不能获得更大的收益这一安定状态的存在。这个问题还没解决，是很火的研究领域。

纳什均衡点

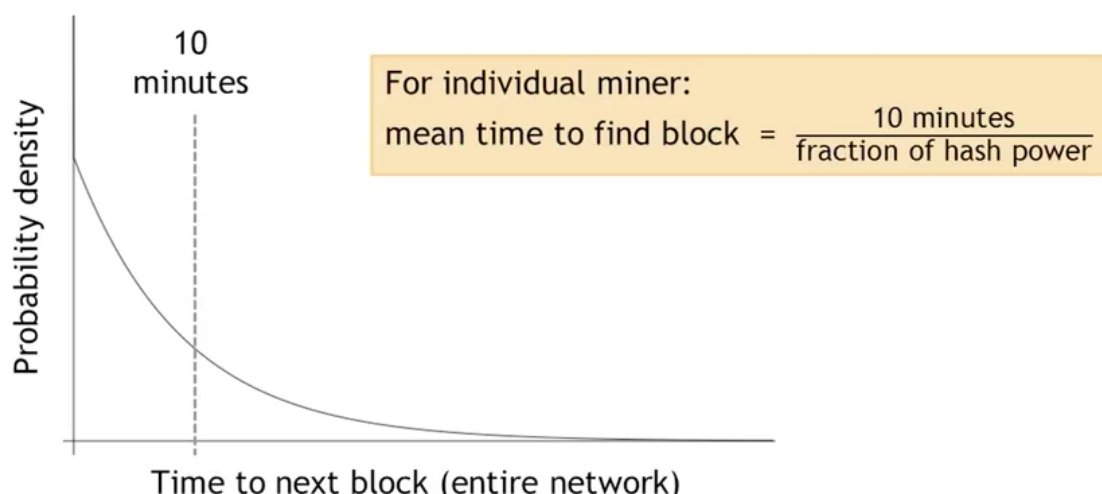
如果某情况下无一参与者可以通过独自行动而增加收益，则此策略组合被称为纳什均衡点。

因为成本函数和工作量证明，我们可以改变安全性的前提。可以不用假设大部分的节点都是健康节点，以哈希算力为权值的节点大多数都遵循着协议，根据下一个区块的提出的竞争性之，自动保证了至少有50%的提出下一个区块的节点是健康节点。

伯努利试验：哈希难题的解决办法只能一个一个试nonce值，谁都没法预测。

泊松过程：事件以一定平均的比例独自发生的连续概率过程。

Solving hash puzzles is probabilistic



如果哈希算力所占比重较小，则不仅每次找到的时间间隔变长，间隔时间的离散程度也越大。

2.4.6 确认简单

nonce值要包含在区块中。

任何节点都可以瞬间确认其他节点产生的区块满足pow属性。

2.5 总体情况

2.5.1 挖掘成本

IF
挖掘报酬>挖掘经费
THEN 挖掘利益上升
where
挖掘报酬=制作区块的报酬+交易手续费
挖掘经费=硬件开销+运用开销 (电气，空调费等)

以上等是存在的问题：

1. 硬件成本固定，但是电费流动成本不可知
2. 挖掘报酬依据找到区块的概率，不仅与硬件有关，还与全局的哈希算力有关
3. 存在汇率问题，成本是美元，报酬是比特币
4. 现在都是以最长有效分支协议挖掘，但是用别的挖掘策略的时候，上式无法判断

参加比特币协议不需要真实世界的身份。谁在任何时候都能制造一个假名密码对。基本上，一个地址指向另一个地址传送命令的信息是通过p2p网络广播的。

比特币的目标是在任何一个节点都能传递全部新的交易和信息，但是p2p网络是不完全的，所以要努力确保信息的中继传递。

系统的安全不仅仅由p2p网络确保，是区块链和所学的所有的东西的集合。

交易包含进区块链，真正地意思是教育获得了足够多的确认。确认数不是固定值，一般取6。

孤立块的形成原因很多，有网络延迟，无效块，二重支付攻击。

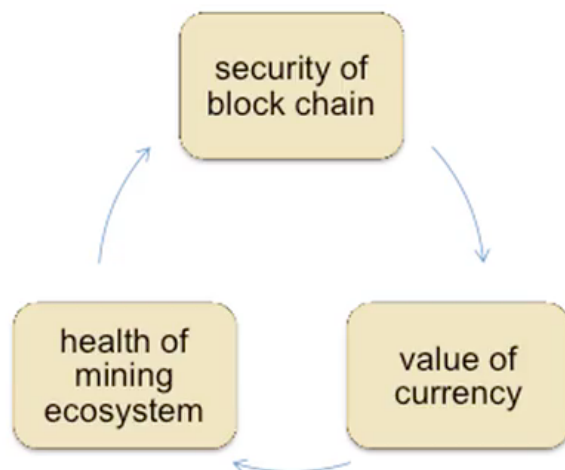
分布式一致性达成渗透入比特币的方方面面，Alice拥有多少比特币的真正意思是，通过比特币p2p网络区块链的记录，合计Alice地址所拥有的比特币。即其他节点认为Alice所拥有的比特币数量。

系统的规则改变，分为硬分叉(hard forks)和软分叉(soft forks)。

2.5.2 自举(bootstrapping) 立ち上げ

区块链的安全性，挖掘系统的健全性，和汇率是相互依存的。


Bitcoin is bootstrapped





比特币从这三特性一个也没有到现在几乎都具备这三个特性，是一个自举过程。

2.5.3 51%攻击



What can a “51% attacker” do?

Steal coins from existing address? 

Suppress some transactions?

- From the block chain 
- From the P2P network 

Change the block reward? 

Destroy confidence in Bitcoin?  

第三章 Mechanics of Bitcoin

回顾：比特币共识

1. 只添加账本
2. 分散化共识达成
3. 矿工验证交易

3.1 bitcoin交易

account based system

缺点：想要确认交易是否有效，必须追溯所有账户的余额。

改善：引入余额管理系统 十分多余

transaction based system

输入：可以多个输入

输出：可以多个输出，交易的剩余部分自己交易给自己

ID：identifier

交易发起人要进行签名

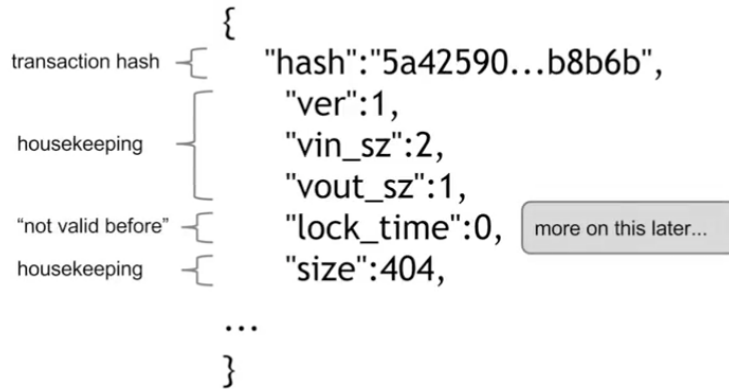
- change address: 零用钱地址 即剩余的比特币要自己交易给自己
- 高效的确认：因为使用了哈希指针所以确认变得十分简单
- 资金统合：多个输入输出

-
- 共同支付(Joint payment)：需要两个支付者的署名

- 交易的构文：

a. metadata：交易的大小，输入数，输出数，交易全体哈希值(ID)，lock_time

The real deal: transaction metadata



b. 输入：

- 参照前一个交易信息，所以必须包含前一个的哈希值，是对前一个交易的哈希指针
- 包含前一个交易的输出参数，必须署名。证明使用了前一个交易的输出值。

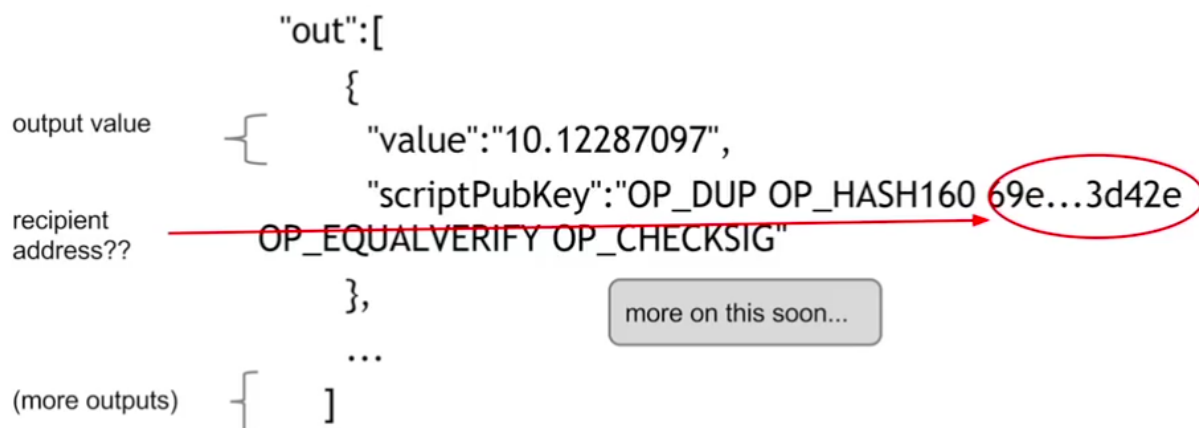
The real deal: transaction inputs



c. 输出：

- 输出值的和必须小于输入值的和
- 差额是手续费
- 接受者的地址 送到特定的公开密钥 实际上是一个脚本 包含了命令与地址

The real deal: transaction outputs



3.2 bitcoin scripts