Nxt 白皮书（中文修订版）

目录

1.Nxt：点对点的电子经济生态系统

1.1 摘要

比特币已经证明了点对点的（对等的）电子货币系统的可行性，并且可以在不需要信托或是中央印钞厂（金融机构）的情况下完成支付过程。但是比特币的一些缺点也使得其成为电子经济的基础变得困难。为了使得整个经济系统能够建立在对等的基础上，必须要做到以下几点：

1.快速的处理数以千计的交易量
2.提供一种产生收入的方法
3.有可行的方法来增加新特征
4.能够在可移动设备上运行

而 Nxt 则满足了以上所有要求，同时还消除了 Bitcoin 的 POW 机制所需要的算力设备军备竞赛。Nxt 是基于 100%的股权证明之上并且需要一种原始的分配方式。但是（很多人）对于这种创始区块的具体分配方式充满了疑惑，他们会问到“怎么解决由于 Nxt 对 73 个大股东的不公平分配而导致的骗局指控？BCNext 这样做出回应：”这个问题是没办法解决的。即使我们有一百万个股东，剩下的 70 亿人仍然会觉得这不公平，物欲横流的世界永远都不会公平（有钱的地方就会有不公平）[1]。”

关于 Nxt 最根本的革新就是透明锻造(Transparent Forging)，这是 Nxt 最核心的改造（创新），使其每秒能处理上千次的交易量。

1.2 简介

随着比特币的出以及市场的接纳，迅速的出现了一些山寨产品，它们简单的改变了比特币的某个方面。让所有的比特币供应者达成共识的难度降低了，共识就是只是简单的制造一种全新的币种。目前大约有 100 多种以比特币资源代码为基础的山寨币。即使与比特币相比，每种山寨币在某些投资环境上可能有其特有的优势，可就其增值以及分裂的特点可能会使得山寨币的成功更令人质疑。此外，也正是因为山寨币是以比特币的源代码为基础的，才会更大限度的延续了比特币的局限性。

为了使交易量增加，一些山寨币仅仅是加快了每种新区块所需要的时间。然而，即便是每分钟一个区块的速度，10 次确认程序也需要将时间拉长至 10 分钟。这对于大量购买是可行的，但是对于购买较小的生活必需品是不实际的。这种方法的弊端之一就在于当降低区块之间的时间间隔时就增加了孤立区块的概率，问题在于在将它们所挖到的区块散播到网络之前，多个节点在对同一个区块进行挖矿。为了测算每分钟上千的交易量需要大量的点对点的带宽或是让交易处理得以集中。前一种办法因为目前特有的网络连接所以不现实，而第二种办法则是与分布式的点对点要求完全背离。只有透明锻造(Transparent Forging)这种解决办法，它能让整个网络来预测哪个节点会锻造下一个区块，从而能够直接的传输交易并且能保证交易中的即时确认，这就消除了网络速度的瓶颈问题。

比特币中经常被忽略的一个缺陷是其作为储藏手段的性质，即像是一个价值不菲的商品。就其本身的特征而言，并没有任何可获利的地方。这看起来仅仅是一个小瑕疵，然而这也正是为什么人们不把他们所有的财富储存在黄金里的原因。如果你有一笔可观数目的财富，把它们都储存在能

产生收入的资产中会更有意义。通过比特币直接赚钱的方法，就是推断它的期货价值（未来价格）。Nxt 有一种革新机制使交易费用得以循环，即用交易费来奖励锻造出目前区块的节点。就目前而言只有三种方式生成交易费：任意消息、Nxt 转移和别名注册，但是随着 Nxt 增加更多的特征，每一个参与锻造的 Nxt 账户所生成的收入也会持续增加。因为 Nxt 拥有一种实用的机制来通过它的投票系统获得更多新的特征，希望社区能同意增加更多有前途的新兴的特征，相应的，也会增加交易费用的数额以及 Nxt 的实际利率。

通过把所有的这些特点与可扩展的框架相结合，Nxt 已经完全成为一个完善且成熟的点对点经济的基础。从微交易，到投资回报产出、公开 IPO、瞬时交易，以及所有 Nxt 所能实现的事情。

1.3 比特币的问题以及 Nxt 的解决办法

1.4 比特币的问题

1.4.1 区块链大小

比特币的区块链是完全按顺序生成的数据区块，它包含了自从比特币在 2009 年 1 月上市以来所有的电子分类账簿。四年之后的 2013 年 1 月，比特币的区块链已经占据了 4 千兆字节(GB)---大约是将一部两个多小时的电影刻在一张 DVD 光盘上所需要的数据量。仅仅在一年之后的 2014 年 1 月，比特币的区块链已经膨胀到 3 个百分点---13 千兆字节(GB)[2]。比特币的区块链正经历着指数倍的增长，因此又必须要修改原始的比特币协议来应对不断增加的区块链。

1.4.2 每天的交易量

在 2013 年年底，比特币的交易量最高达到了每天 70,000 的峰值，或是平均每秒一个交易量(TPS)。目前的比特币标准区块在一兆字节，是由称之为“全节点”的客户端每隔十分钟产生的，将目前比特币网络总承受量局限在最大 7TPS。把这个与 VISA 的网络工作相比，它的承受能力可以达到 10000 TPS，你就会发现比特币不能与如今所存在的相抗衡。

增加比特币系统的公共使用量会使比特币很快达到每日交易量限额，并且会阻断其更多的增长。为了防止这种情况的发生，比特币软件开发者正在研发一种“lite node”客户端，声称能够简化支付验证节点而不会被区块的尺寸所限制。为了在相同的平均十分钟里能操控更多的生产量，SPV lite node 不会在他们处理的区块里进行全面的安检，相代替的是对那种竞争的矿工的散列且多样的区块链进行检查，并假定由大多数矿工所生产区块链版本是正确的。用比特币麦凯恩的话来说就是“没有确认所有东西的真伪，仅仅是因为 SPV 信任大多数的矿工是诚实的。只要大多数是诚实的，SPV 就能工作，但全节点确实提供了更好的安全性。例如，你开了个在线店，你就会感觉到运行全节点是有意义的。[3]

1.4.3 交易确认时间

2013 年的大部分时候比特币的交易确认时间是 5 到 10 分钟。自从中国的银行不许再受理任何比特币的公告一出，比特币的平均交易时间就明显的增加到 8 到 13 分钟，以后的交易时间每时每刻都充满着不确定因素。中国的银行预算比特币一天可以产生大约 650,000 的交易量并且交易确认的时间最长达到 20 多分钟。因为最终确定比特币的交易需要很多的认证，因此在用比特币购买资产的过程结束之前会很耗时。

1.4.4 中心化的疑虑

比特币难度的增加[4]以及网络 hashrate [5]的结合成为新手们进入的很大阻力，并且给现存的矿机减少了利润。Bitcoin 所采用的区块奖励机制驱动了创建更大、更专业的挖矿设备[6]，以及对大型矿池的依赖[7]。这已经导致了"中心化"的结果，因为越来越少的人控制着越来越多的算力。这不仅导致了 Bitcoin 本身设计时要避开的这种算力结构，而且表明单个挖矿设备或者矿池有能力占据网络总算力的 51%并且实施 51%攻击。只需要 25%总的网络 hashpower 的攻击也是存在的[8]。
2014 年 1 月初，GHAsh.io 由于其矿池算力接近 51%而不得不减少其矿池算力[9]。几天之后，该矿池的算力减少至整个网络算力的 34%，但是随后算力马上又增加了。1 月份第三周，Bitcoin 两个最大的矿池的算力已经达到了网络总算力的 60%了[10]。

1.4.5 工作量机制对资源的消耗

现有比特币的交易确认以及创造新的比特币用于流通需要大量的计算算力持续的操作。这种电脑算力是由那些“矿工”们通过所谓的矿机进行操作的。比特币矿工们相互竞争来为整个比特币的区块链增加下一个新的交易区块。这是通过”哈希值“来完成的---将过去十分钟内所有发生的 Bitcoin 的交易捆绑，并尝试将它们编译到一个区块数据中，而这个区块含有一个特定数目的连续 0 的随机数。绝大多数旷工哈希计算所产生的区块都不包含该目标 0 值，因此它们需要作出轻微的改变并继续尝试。为了成功找到区块而进行的大量的尝试称之为"gigahash"，也即挖矿设备是用每秒钟能够进行多少 gigasashes 来分级的，表示为"GH/sec"。第一个产生准确区块的旷工将会收到 25 个新的 Bitcoins 的奖励，按照目前的价格来计算为$25,000。这些在旷工之间为了获得奖励的竞争，将会每隔 10 分钟进行一次又一次的重复。到 2014 年初，每天产生的作为奖励的 Bitcoin 的价值在 350 万美金。
看到如此多的奖励，旷工之间展开了激烈的军备竞赛，以便增加获得奖励的概率。起初，Bitcoin 是用普通计算机的中央处理器(CPU)来挖矿的。后来中门的图形处理单元(GPU)也被用于增加速度。再后来是现场可编程门阵列 (FPGA) ，紧接着则是专用集成线路 (ASIC)芯片。 ASIC 技术是 Bitcoin 旷工的顶级技术，然而军备竞赛则产生了各种各样的 ASIC 芯片。目前的 ASIC 芯片是 65nm 单元的，是基于纳米级的晶体管。它们被 2014 年初的 28nm 的 ASIC 和 2014 年中的 20nm 的 ASIC 所替代了。其中的一个例子是 Butterfly Labs"Monarch" 28nm 的 ASIC 芯片，它能提供的算力为 600GH/sec，电力消耗为 350 watts，价格为单片 $2100。Hashblaste 刚出来的预定芯片则含有 3 个 20nm 的 ASIC 芯片，能提供的算力为 3300 GH/sec，能耗为 1800 watts。

大多数的挖矿设备都将在 2014 年中旬升级到该性能标准。
目前支撑 Bitcoin 运行的挖矿设备构架是惊人的。Bitcoin 的 ASIC 是愚蠢的---它们只能进行 Bitcoin 的区块计算，并无其他用途，但他们能够以超级计算机的速度进行那种计算。2013 年 11 月，福布斯杂志撰文写道 "全球 Bitcoin 的算力是 500 个最顶级超级计算机联合算力的 256 倍 "[11]。2014 年 1 月中旬,blockchain.info 显示支撑 Bitcoin 运行素需要的持续算力为 18 million GH/sec。在一天 86,400 秒内，这意味着旷工们为了找到能够获得 350 万美金奖励的区块，将会有大约 1.5 万亿的尝试区块被生成和拒。因此大约有 99.99999999 % 的 Bitcoin 的算力并没有用于治疗 DNA 模型的癌症或者 E.T 的无线电研究---相反，他们被完全的浪费了。
旷工为了支持 Bitcoin 所小号的电力和成本是巨大的。如果所有的 bitcoin 的挖矿设备都拥有 "Monarch" 级别的能力--除非它们升级，否则不会达到这样的级别---这 30,000 个设备将耗资 63,000,000 美金，而且每天持续消耗的电力超过 10 兆瓦，也即是每天消耗的电费好过 3,500,000 美金[12]。而真实的数目比目前的还要大，因为支持 Bitcoin 运行的挖矿设备相对比较低效。随着 Bitcoin 从目前的一秒钟一个交易增大至最大一秒钟 7 个交易，这些数字也正以指数倍数在向上增长。

1.4.6 持有工作量机制虚拟币的资源耗费

除了大规模的电力消耗之外，单纯的持有比特币也有一些隐性支出。对于每个区块资金，生成区块的实体都需要固定的津贴。在写这篇文章的时候，津贴是 25 个比特币，为今年比特币的总供给量提供了 10%的膨胀率。对于持有价值 1000 美元的比特币有些人而言，今年需要花费 100 美元一个的比特币“支付”给矿工们，来保证网络的安全运转。

1.5 Nxt 的解决办法

1.5.1 密码学基础

Nxt 的主要交易是基于 curve 25519 算法，该算法使用快速、高效、安全的椭圆曲线秘钥算法 (elliptic-curve Diffie-Hellman function[13])生成了共享秘钥。这种算法最先是由 Daniel J. Bernstein 在 2006 年提出的[14]。
Nxt 中的信息签名是使用 EC-KCDSA(elliptic-curve Diffie-Hellman function)来实现的，该算法是 IEEE P1363a 的一部分，是由 the KCDSA Task Force team 在 1998 年提出的[15]。
所有的算法都是基于平衡速度和安全来选择的，其秘钥长度为 32 字节。

1.5.1.1 加密算法

当爱丽丝给鲍勃发送了一份加密文档时，她：

1 Calculates a shared secret:

```
1. shared_secret = Curve25519(Alice_private_key,Bob_public_key)
```

2 Calculates N seeds:

```
1. seedn = SHA256(seedn-1), whereseed0 = SHA256(shared_secret)
```

3 Calculates N keys:

```
1. keyn = SHA256(Inv(seedn)),where Inv(X) is inversion of all bits of X
```

4 Encrypts the plaintext:

```
1. ciphertext[n] = plaintext[n] XOR keyn
```

Upon receipt Bobdecrypts the ciphertext:

5 Calculates a shared secret:

```
1. shared_secret = Curve25519(Bob_private_key,Alice_public_key)
```

6 Calculates N seeds (identical to Alicestep):

```
1. seedn = SHA256(seedn-1),where seed0 = SHA256(shared_secret)
```

7 Calculates N keys (identical to Alicestep):

```
1. keyn = SHA256(Inv(seedn)),where Inv(X) is inversion of all bits of X
```

8 Decrypts the ciphertext:

```
1. plaintext[n] = ciphertext[n] XOR keyn
```

注意：如果有人猜到文档的一部分内容，他们可以通过使用相同的密匙对来破译爱丽丝和鲍勃之间的信息。所以，建议分别为每一次的交流建造一对私钥和公钥。

1.5.2 区块链

同其他加密的货币一样，NXT 交易的总账是建立和储存在一系列的区块里的，也就是所谓的区块链。每个区块链的备份都存放在 Nxt 网络的每个节点里，而且在每个节点上没有加密的每个账户都能够生成区块，只要至少一个新入账户的交易已经确认了 1440 次。任何账户只要达到了这个标准就会被视为"激活账户"。

在 Nxt 里，每个区块都包含着 255 个交易量，每个交易都是由包含识别参数的 192 字节的数据头开始的。一个区块里的每个交易量都是是由 128 个字节所代表着。总共加在一起就意味着最大的区块大小有 32K 字节。所有的区块都包含以下参数：

●一个区块版本
●一个区块时间戳，从源区块开始的用秒来计算
●之前区块的 ID
●区块里所储存的交易数目
●区块中总的 Nxt 交易量
●区块中总的交易费用

●区块的负载长度
●区块负载长度的散列值
●生成区块的账户公匙
●区块的生成签名
●整个区块的签名

每个链条上的区块都有一个"生成签名"的参数。激活账户用自己的私钥在原先的区块上签署“生成签名”。这就产生了一个 64 字节的签名，之后通过 SHA256 散列该签名。哈希产生的前八个字节给出了一个数字，作为一个“hit”。这个“hit”与目前的目标值（是一个 64bit 的数字）相比较。如果计算出的“hit”值要比“目标值”低，那么就可以生成下一个区块了。

因此产生了“Proof of stake”的算法，因为对于每个激活账户来讲，“目标值”都是与它自身所确认的余额成比例的。一个持有 1000 个币的账户得到的目标值是持有 200 个币账户所得到目标值的 50 倍。因此，拥有 1000 个币的持有者产生的区块数是持有 20 个币的人产生的 50 多倍（从长远角度来说）。

“目标”值并不是固定的。随着先前区块的时间戳的流逝，每秒钟都在增长。如果在最初的一秒钟内没有哪个账户的“hit”值是低于“目标”值的，则下一秒钟目标值就会翻。“目标”值会持续的翻倍直到一个活动账户的“hit”值有一个较低的数值。还有一个“基本目标”值会以 60 秒的间隔设定为目标值。正是因为这个原因，一个区块，平均产生的时间会在 60 秒。

即使在网络上只有很少的激活账户，他们其中的一个最终会产生一个区块因为“目标”值会变得相当大。通过将你账户的“hit”值与目前的“目标”值相比，你就可以估算出你的“hit”值还有多久能成功。

当一个激活账户赢得产生区块的权利时，就能将任何可获得的且未确认的交易放入区块中，并用所有需要的参数来填充该区块。然后这个区块就会被传播到网络中作为一个区块链的备选。

每一个区块中的负载值、“hit”、产生的账户以及签名都能被网络上接收到它的节点所确认。每个区块参考之前的区块，区块形成的区块链可以用来追溯和查询网络中素有的交易历史，所有这些都会追溯到创世源区。

1.5.3 交易

计算每个 Nxt 账户的余额需要对整个区块链进行扫描。尽管这听起来效率很低，但是就目前网络与 CPU 的速度而言，这并不是一个很大的计算量。处理这些工作需要 Nxt 服务器因此也就允许了更低能耗的移动设备成为 Nxt 的节点。

Nxt 交易的细节如下所示：

1.发送者指定了交易的参数。交易的种类有很多（发送钱币，创建别名，发送信息，发行资产或对资产下订单）但是任何交易的几个参数都需要指定。

- ●发送账户的密码
- ●交易的费用
- ●交易的截止期限
- ●随意的“参考”交易

2.所有交易的输入值都要通过检查。比如：强制性的参数必须指定：交易费不少于零，交易截止日期不少于一分钟。

3.如果参数核实的结果不出现意外的话：

①.通过所提供的密码来计算产生账户的公钥。

②.产生账户的账户信息可恢复，并且交易参数要进一步的验证。

●发送的账户的余额不能为零

●发送账户的确认余额不能低于交易额与交易费用的总和

③.如果交易账户有足够的资金提供给交易额

1.产生一个新的交易，其类型与子类型值要设定为与已经产生的交易的类型相匹配（发送钱币，创建别名，发送信息等等）所有指定的参数都包含在交易对象中。唯一的交易 ID 也是随着对象的创建而生产的。

2.交易是用发送账户的密匙所签署的。

3.加密的交易数据被放置在信息里，信息用于指导网络节点来处理交易。

4.交易被传送到网络上的所有节点。

4.服务器会反馈一个结果代码：交易 ID，如果交易成功的话；如果参数确认失败，则反馈错误代码和错误信息。

1.5.4 交易确认

所有 Nxt 的交易都被认为是"未确认的"，除非它们已经被包含在有效的网络区块中。新建立的区块会通过创建他们的账户分散到网络中。而且包含在区块里的交易就会得到确认。因为随后的区块会添加到现有的区块链，因此每增加一个区块就会对现有的交易进行增加一次确认。

在经过 10 次确认之后，Nxt 交易被认为是可信的。如果出现问题，网络有可能重新组织最近的 720 区块，所以一个交易在 721 次的确认之后是不可逆的。而已经被确认了 1440 次的交易则被认为是永恒交易。

1.5.5 股权证明 Proof-of-Stake

在以前陈旧的 Proof-of-work 模型中，网络安全是由节点通过“工作”来保证的，他们借用他们的资源（电脑/处理时间）来帮助加强网络，并且阻止恶意袭击。这些节点因为他们的“工作”而被奖励了一些区块的币，这些数量以及他们出现持续的时间都基于特定的网络。这种办法的弊端就是需要越来越多的时间处理（以及持续的能量）因为随着时间的流逝，指定的节点来支撑网络的运转就格外重要。

换句话说，随着网络越来越快的发展，单个节点来支撑网络的积极性就越来越少，因为他们潜在的奖金都被越来越多的节点所划分。一些节点用专业的、专有的和昂贵的硬件持续的增加资源投资，并且增加能量消耗。随着时间的推移，很讽刺的是，网络将会越来越中心化，较小的节点（工作量很小的节点）会退出，因为他们的奖金会流向更大的节点（那些能负担的起更多资源及能量的节点）。

说到这点，最近 GHash.io 矿池的算力已经非常接近比特币算力的 51%。这样的话，单个个体已经控制了区块链，去中心化的概念就完全消失了[16]。

在 Nxt 所运用的 proof-of-stake 模型中，网络的安全是由拥有"股份"的节点所维护的。

1.5.6 网络

Nxt 的网络是由节点组成的。节点在本质上来说是任何贡献于网络的设备。任何运行 Nxt 的 NRS 客户端(Nxt Reference Software)的设备都是一个节点，并且因为源代码可以开发成本土的客户端，它们也会成为节点。节点可以分为两种类型：“有标记的”和普通的。每个标记过的节点都继承了基于标记账户所持有 Nxt 数目的权重，可以只有 1Nxt，或者，五百万到一千万的 Nxt，.是没有上限的。拥有权重越大的标记节点，其可信度也越高。
如果一个攻击者想要标记一个节点从而获得网络的可信度，然后用这种可信度去进行攻击，进入的阻碍（消耗 Nxt）就会限制这些滥用。一旦投票系统得以实施，其它节点可以发起投票来禁止或是惩罚网络上的恶意节点。

1.5.7 透明锻造

为了了解透明锻造，首先就要理解锻造本身的过程。对于激活的锻造账户，其锻造到区块的机会与它所持有的 Nxt 数目以及网络上所有激活的锻造 Nxt 数目是成比例的。也需要一定的随机性来消除对相对靠后的已知的锻造者的攻击，但是，对于相对靠前的而言，就需要尽可能的准确以减少对网络带宽的使用。这些明显自相矛盾的需求是通过以下 2014 年 1 月 3 号发布的源代码而得以实现：

```
1.  Account account = unlockedAccountEntry.getKey();
2.      User user = unlockedAccountEntry.getValue();
3.      Block lastBlock = Block.getLastBlock();
4.      if ( lastBlocks.get(account) != lastBlock )
5.      {
6.          byte[] generationSignature =
    Crypto.sign(lastBlock.generationSignature, user.secretPhrase);
7.          byte[] generationSignatureHash =
    MessageDigest.getInstance("SHA-256").digest(generationSignature);
8.          BigInteger hit = new BigInteger(1, new byte[]
    {generationSignatureHash[7], generationSignatureHash[6],
    generationSignatureHash[5], generationSignatureHash[4],
    generationSignatureHash[3], generationSignatureHash[2],
    generationSignatureHash[1], generationSignatureHash[0]});
9.          lastBlocks.put(account, lastBlock);
```

```
10.         hits.put(account, hit);    // jl777: hit now contains a
    deterministic but pseudo-random number
11.         JSONObject response = new JSONObject();
12.         response.put("response", "setBlockGenerationDeadline");
13.         response.put("deadline",
    hit.divide(BigInteger.valueOf(Block.getBaseTarget()).multiply(BigIntege
    r.valueOf(account.getEffectiveBalance()))).longValue() -
    (getEpochTime(System.currentTimeMillis()) - lastBlock.timestamp));
14.         user.send(response);
15.     }
16.     int elapsedTime = getEpochTime(System.currentTimeMillis()) -
    lastBlock.timestamp;
17.     if ( elapsedTime > 0 )
18.     {
19.         BigInteger target =
    BigInteger.valueOf(Block.getBaseTarget()).multiply(BigInteger.valueOf(a
    ccount.getEffectiveBalance())).multiply(BigInteger.valueOf(elapsedTime)
    );   // jl777: chance proportional to effective balance
20.         if ( hits.get(account).compareTo(target) < 0 ) {    // jl777: as time
    elapses the target gets larger, eventually triggering the acct closest to
    target.
21.             account.generateBlock(user.secretPhrase);
22.         }
23.     }
```

大拇指法则决定了一个账户每天能锻造到的区块量是(账户余额/1000000000)*1440，以上的前提假设是：所有的 Nxt 都在进行锻造，以及一天产生 1400 个区块。然而这两个数据一天内的变化是很大的。

由于锻造机会是计算出来的，因此就能够预计出那个账号锻造出下一个区块以及什么时候锻造出来。因为 hit 值已经确定，拥有多账户的人可以计算出哪个账户最有可能锻造出下一个区块，因此可以将所有的 Nxt 转移到那个账号里。这就是为什么要选择有效余额而不是实际余额。一个账号存入资金时的时间延迟以及资金转移时的时间延迟都会减少 Nxt 受攻击的有效数目。

通过储存来自所有账户的 hit 值，如果每个节点都知道哪个账户是处于激活锻造的，就有可能让所有的节点预测哪个账号会锻造出最近的区块。

因为每个节点都有基于可视节点和激活锻造账户变化的不同网络拓扑结构，所以并不是 100%的准确，但这是事先的设计。当然也需要一些错误的因素来阻止攻击者通过计算出最近的区块锻造者而对 Nxt 网络实施攻击。只要预测准确率接近 100%，网络拥堵问题就极大地减少，从而允许数以千计的实时交易。

透明锻造允许在去中心化的网络中存在中心化的操作。这是 Nxt 最基本的突破。透明锻造允许每个用户客户端自动决定谁将产生下一个区块，然后他们就可以将把他们的交易发送至那个节点上。为了实现即时交易，允许额外的费用。透明锻造另一个同等重要的特点是协议杰出的安全性，它可以临时性的将产生一下个区块的节点的锻造能力减少至 0.。这个特性设计用于阻止拥有 90%Nxt 数目的节点进行分支或者分叉。因此，如果一个节点拥有 90%的 Nxt 而未按照计划产生区块，系统就会临时的将其锻造能力减少到 0 以阻止可能的分叉。它的锻造能力就会分配给网络中剩余的节点，因此网络的能力还是 100%，所以，不管该潜在的对手它在其它分支上做什么，都会被高级的共识机制（还未披露）所抵消。

透明锻造意味着什么？它意味着每个人都可以预测（很大几率）谁以及什么时候会产生下一个区块。这就给予了 Nxt 很多优势：

1.交易能够直接发送给即将产生下一个区块的锻造者（如果他愿意披露它在网络上的地址），因此节省了交易流量并很快接近 VISA / MasterCard 的交易量。

2.区块可以提前产生，并且在它们生效（时间戳生效）之前发送给大多数的锻造者，因此很大程度上就减少了孤立区块的概率。

3.由于可以预测未来区块时间（区块速率），因此设定适当的费用来确保重要交易能够快速确认就变得有可能（不用在一个区块中花费太多）。

4.也许最重要的是，网络能够检测出哪个锻造者没有参与到区块生成中来并采取相应的措施。

作为 100%的 POS 货币，Nxt 能够预防政府和可以获得很多 ASIC 的财团，而且有了透明锻造特征，甚至可以预防某些人购买绝大多数的币。

所以到底是什么让 Nxt 能成为下一代货币？并不是那些漂亮的特征，比如去中心化的交易、DNS 或是去中心化的应用商店，而是透明锻造机制（促成了这一点），而且这只是 BCNext 计划的第一部分。

1.5.8 交易费用

交易费是 Nxt 如何再循环至网络的途径。现有的交易资金是通过发送 Nxt、创建别名或者发送信息来产生的。

交易费用目前设定为每次支付最少 1 个 Nxt，并且直到交易数目填满一个区块，1 个 Nxt 的费用将足够将其包括在区块内。而且随着 Nxt 价格的增加，最小的交易费用将下调至用户可以接受的程度。比特币的交易费为 0.0001 的 BTC，随着比特币价格的增加交易费将会变得越来越不实际。随着 Nxt 价格的上涨，交易费也会逐渐降低，Nxt 将会非常适合于微支付。到那时候，即使需要更小的单位，比如 milli-NXT, micro-NXT 甚至 femto-NXT，我们可以发行彩色币来表示任何内容。

1.5.9 回收磁盘空间

区块链的膨胀是一件大事，它与任何的加密货币相关，尤其是对那些像 Nxt 交易量很大的来说。通常，添加到区块链的操作会按照他们使用的空间大小来收费，这也是为了限制节点故意膨胀区块链。
但是，随着时间的流逝，重新从源区块开始计算所有内容是很没有效率的。Nxt 计划进行每年的检验点，它会为所有的节点创建一个起点去使用，其频率可以让 Nxt 的股东们进行投票表决。通过使用电子签名，可以确保每年检验点的有效性。拥有资源越多的网络节点（比如专用服务器）也能继续支撑整个区块链，并且作为服务提供商而受到奖励。（这是我的拙见）
举个例子，到 2012 年中期，比特币的区块链仍然保持在 1GB 大小内。而现在，仅仅一年半之后（2014 年的 1 月）随着比特币日趋增长的流行以及越来越多的交易，区块链已经膨胀到了将近 13GB 的大小。很明显，对于绝大多数的设备来说提供这么大容量的区块链是不可行的。即使是整个链条的传送就要花费好多小时，而且还要取决于网络连接速度。
查看：https://blockchain.info/charts/blocks-size

1.5.10 设备可携带性

由于它是 Java 代码，POS 哈希散列以及能够修剪和减少区块，Nxt 非常适合于运行在小巧，低功率以及低功耗的设备上。安卓以及苹果的应用也正在开发中，而且 Nxt 客户端 NRS 已经运行在低功率的 ARM 设备上了，比如 RaspberryPi。
将 Nxt 应用于低功率或者联网设备是很容易的，比如说，智能手机。这些设备支持大部分网络。因为全球数以百万计的人已经拥有了智能手机，Nxt 能够在这些设备中快速的得到应用从而来支持网络，而不用像传统密码学货币那样花费很多钱。
Nxt 的特征，比如说瞬时交易使得智能手机成为一个理想的平台在日常消费中来使用 Nxt（食品，燃料等等）。在这个领域，其它密码学货币也存在解决方法（比如，比特币），但因为需要大量的资源（算力）来维护网络，因此使用这些设备并不能对网络健康或稳定性带来帮助。而对于 Nxt，任何有发送和接受交易功能且有一定算力的设备都能增加网络的稳定性以及去中心化。

1.6.0 锻造及 Nxt 的产出

锻造一个区块的机会取决于基础目标值 BaseTarget（对所有人都一样）、从上一区块开始的时间 TimeSinceLastBlock（对所有人都一样）以及你的账户余额 Balance。
Code：机会 = 基础目标值 *从上一区块开始的时间 * 账户余额

1.6.1 Nxt 锻造计算

作者： mthcl http://www.docdroid.net/9yub/forging.pdf.html
我们从概率论的观点来讨论 Nxt 的锻造机制，通过公估算出几个重要的参数，比如一个账户锻造到区块的可能性，一个账户锻造到最长序列连续区块的长度，以及目前区块链胜过其它的可能性。
结论：

●如果一个账户持有总有效余额 b 比例的数目，而其它账户的锻造力相对较小，那么它产生一下个区块的可能性是由(4)来决定的。
●拥有比例为 b，则大约为 $b + b^2$，例如，区块产生的可能性与有效余额成二次方比例。
●很容易得出账户 k 锻造出区块的可能性为 $b_K(1 - f(b))$
讨论：https://forums.nxtcrypto.org/viewtopic.php?f=17&t=836

1.7 计算

详细内容：from
bloodierookie https://bitcointalk.org/index.php?topic=397183.msg4645132#msg4645132
1) Mathematical background:

a) Groups: I recommend to read the wiki article on groups (http://en.wikipedia.org/wiki/Group_%28mathematics%29) at least partially. If you don't like to read it for whatever reason and still want to receive some impression of what a group is, here is the short (and not mathematical strict) version: You know the set of integers. There is an addition defined on the set of integers. You can do calculations like a + (b + (-a)) = a + ((-a) + b) = (a + (-a)) + b = 0 + b = b If you have a set of elements and you can do calculations with them like the one above, the set is called an (additive) commutative group. There always is a neutral element, denoted by 0, which you can add to any other element without changing the element. To every element a there is an inverse element, which in the case of an additive group is denoted by -a. Adding those 2 elements together results in the neutral element: a + (-a) = 0. How about multiplication? Does the set of integers form a group with respect to multiplication as operation? No, because there is no inverse element to most integers with respect to multiplication, e.g. there is no integer a that satifies 2*a=1 (1 is the neutral element of multiplication). How about the rational numbers Q and multiplication? Well, almost! The only element not having an inverse is the 0. But Q without the element 0, i.e. 0 Q\{0}, is a group with respect to multiplication. In a multiplicative group the inverse element of a is usually denoted by a^-1 and the neutral element by 1.

b) Modular arithmetic: I recomment to read the wiki article on modular arithmetic (http://en.wikipedia.org/wiki/Modular_arithmetic) at least partially. Again, here is the short version: When you were young, you didn't know about fractions. So if you had 7 apples and wanted to split them among 3 children, each child would get 2 apples and there was 1 apple remaining: 7 = 2*3 + 1. In modular arithmetics you would say "7 is congruent 1 modulo 3" and write 7 ≡ 1 mod 3. The neat thing about remainders of divisions is that you can calculate with them as if they were numbers: 7 ≡ 1 mod 3 and 8 ≡ 2 mod 3 and therefore 7+8 ≡ (1+2) mod 3 ≡ 0 mod 3, 7*8 ≡ (1*2) mod 3 ≡ 2 mod 3 When you do calculations modulo a prime p, the set of remainders forms a commutative group with respect to additon and the the set of remainders excluding the 0 forms a commutative group with respect to multiplication. With those 2 operations (and some additional law

which I omit for brevity), the set forms the finite field Fp.

c) Montgomery curves: I recomment to read the wiki article on elliptic curves (http://en.wikipedia.org/wiki/Elliptic_curves) and the article on Montgomery curves (http://en.wikipedia.org/wiki/Montgomery_curve) at least partially. Here is the short version: Elliptic curves are described by equations which have the form y^2 = x^3 + ax^2 + bx + c with some numbers a,b,c. (Usually the second highest term is eliminated by a suitable translation of x). Any point on the curve can be represented by its affine coordinates (x,y). It is possible to define an addition for points on the curve (no, not by simply adding the x and y coordinates!) Read the wiki article for details. With this addition (and by introducing another point called the point at infinity which serves as the neutral element), the set of points form an additive, commutative group.The addition of points gives rise to another operation: if you add a point P to itself giving you P+P, you could as well write 2*P as a short version of P+P. You can guess what 3*P, 4*P, and so on means. The product of a point by a number is called scalar product. One can also restrict the allowed values for x (and thus for y) to finite fields Fp. Montgomery curves have the form By^2 = x^3 + Ax^2 + x. The reason why those curves are interesting for cryptography is that they allow fast arithmetic. Switching to projective coordinates X and Z (x=X/Z) one can add points P=(Xp:Zp) and Q(Xq:Zq) without knowing the y coordinate as long as the X and Z coordinates of P-Q are known. However, if you only know the projective coordinates of one point P=(Xp:Zp) it is impossible to recover the y coordinate. You need to know for instance the projective coordinates of P, Q and P+Q to recover the y coordinate of P.

2) Analyzing and debugging the process of signing and verification: For the rest of post we fix p=2^255-19 and q=number of elements of the group generated by the so called base point G of curve25519. p and q are both prime numbers. The Montgomery curve is given by the equation y^2 = x^3 + A*x^2 + x where A = 486662. Let's start by looking at the processes at a high level, meaning that for the moment we believe that the software is actually doing what the comments of the author suggest. The sign method of the Crypto class, which has a message and a secret phrase as input, first calls keygen(P, s, k=SHA256(secret phrase)) where k is the only [in]-argument and P and s are [out]-arguments. keygen() clamps k and then calls core(). core() calculates the x-coordinate of P=k*G (which is the public key for k). Since s ist not null, the calculation continues. First, the y-coordinate of the point P is recovered and then s is calculated as k^-1 mod q if the y-coordinate is positive or (-k)^-1 if the y- coordinate is negative (we will see the reason for distinguishing those 2 cases in a moment). One could say s = (sign(P) * k)^-1 if sign(P) is defined to be the sign of the y-coordinate of P. The Crypto class then calculates x = SHA256(message + s) (the "+" means the arrays are appended rather than added), Y = public key of x (with a call to keygen(Y, null, x)) and h = SHA256(message + Y) and finally calls the sign method of the Curve25519 class which returns v = (x-h)*s mod q. The signature is defined as pair (v,h).
The process of verifying is as follows: v and h are recovered from the signature and then Curve25519.verify(Y, v, h, P=publik key of k) is called in order to recover the same Y as

above. This is tested comparing h2 = SHA256(message + Y) with the given h. Let's take a closer look how Y is recovered: For the verification to succeed we need to calculate Y = v * sign(P) * P + h * G since then we have Y = (x - h) * s * sign(P) * P + h * G = (x - h) * (sign(P) * k)^-1 * sign(P) * k * G + h * G = (x - h) * G + h * G = x * G = public key of x (which is exactly the definition of Y in the sign method). Oh no! We need to calculate sign(P) but we can't recover the y-coordinate of P! Fear not, a neat trick will help us. If we knew that y is the y-coordinate of P, then the x-coordinates of P+G and P-G can be calculated to (the x denotes the x-coordinate, y the y-coordinate) (P+G)x = (Py^2 + Gy^2 - 2 Py Gy)/(Px - Gx)^2 - Px - Gx - A =: s[0] (P-G)x = (Py^2 + Gy^2 + 2 Py Gy)/(Px - Gx)^2 - Px - Gx - A =: s[1] With the values for s[0] and s[1] given, the rest of the code of the verify method would calculate v*P+h*G by applying an addition chain. Calculating Py^2 is no problem since we know the curve equation. There also is a method recip() for calculating a root for Py^2. Let's say we switch the definition of s[0] and s[1] whenever the root returned from recip is negativ. So in that case we would have s[0] := (P-G)x = (-((-P)+G))x = ((-P)+G)x s[1] := (P+G)x = (-((-P)-G))x = ((-P)-G)x where we used that negating a point is a reflexion across the x-axis and thus doesn't change the x-coordinate. Switching the definition of s[0] and s[1] means we simply switch from P to -P in the calculation . But we might have used the wrong y-coordinate of P! So let's check what the influence of that is: We have to distinguish 4 cases (real y-coordinate denoted by y, y-coordinate returned from recip() denoted by y'): i) sign(y)=1 and sign(y')=1 ==> y'=y. Thus s[0] = (P+G)x = (sign(P)*P+G)x, s[1] = (P-G)x = (sign(p)*P-G)x ii) sign(y)=1 and sign(y')=-1 ==> y'=-y. Looking at the formula for P+G and P-G we see that P+G becomes P-G when replacing y with -y and P-G becomes P+G. Since in the case sign(y')=-1 we switch s[0] and s[1] as well we end up with s[0] = P+G = (sign(P)*P+G)x and s[1]=(sign(P)*P-G)x as in case i). iii) sign(y)=-1 and sign(y')=1 ==> y'=-y. We don't switch s[0] and s[1] in this case but we really calculate s[0]=(P-G)x, s[1]=(P+G)x because we switch from y to -y in the formula. So s[0] = (P-G)x = ((-P)+G)x = (sign(P)*P+G)x, s[1] = (P+G)x = ((-P)-G)x = (sign(P)*P-G)x iv) sign(y)=-1 and sign(y')=-1 ==> y'=y. We switch s[0] and s[1] resulting again in s[0] = (P-G)x = ((-P)+G)x = (sign(P)*P+G)x, s[1] = (P+G)x = ((-P)-G)x = (sign(P)*P-G)x. So in all the cases we have s[0] = (sign(P)*P+G)x and s[1] = (sign(P)*P-G)x and the addition chain will output v * sign(P) * P + h * G just as desired! This means the algorithm for signing and verifying is ok and verify() should return true every single time when we use the signature returned from sign(). And still verification fails every now and then, so there have to be bugs somewhere! Maybe a bug in the addition chain? Such a chain is hard to debug, it's simpler to write our own methods for scalar multiplication and addition of points. I did that and it turns out that my method returns the very same Y as the addition chain every time. It seems the addition chain is working, the bugs must be somewhere else in the code.

The next part of the code which I took a close look at was the calculation of s = (sign(p)*k)^-1 in the core() method. After the point P=k*G is calculated using a Montgomery ladder (I have checked the code for adding points and doubling a point), the y-coordinate of P is recovered. How that? Well the ladder not only gives us P but also P+G. Since G is known too the y-coordinate can be calculated. I have checked the derivation of the formula for y, it's valid. sign(P) is ok, what about the calculation of k^-1, how to do that?

Calculating the inverse k^-1 of k mod q can be done in 2 ways: i) Fermat's little theorem states that for a prime q and any integer a the following equation holds: a^q ≡ a mod q. Multiplying with a^-2 on both sides gives a^(q-2) ≡ a^-1 mod q. We simply have to calculate a power of a. This can be done similar to the Montgomery ladder giving us a time independent algorithm. ii) We can use the extended Euclidean algorithm. This algorithm computes the gcd (greatest common divisor) of 2 integers a and b. As a byproduct it can also output the linear representation of the gcd: gcd(a,b) = t1*a + t2*b. That is nice because if we choose a=k and b=q and compute 1=gcd(k,q) = t1*k + t2*q then we have after taking the equation mod q: t1*k ≡ 1 mod q which means t1=k^-1. The author of the Curve25519 class chose the latter approach. It might be a problem because the algorithm is time dependent on the input und thus not safe against timing attacks. I checked the algorithm and it appeared to be ok so I plugged in some numbers. Testing different values for k and validating that the returned value is indeed the inverse of k, I gained confidence in that method. The only thing you should not do is setting k to a negative number like -1. It is interpreted by the algorithm as 2^256 - 1 (positive number) and will not return q-1 as it should. Note also that if egcd32() returns a negative value for s in the method core(), then q is added to s to make it a positive number.

There is only one method left, that is Curve25519.sign(), so the bug must inside that method. Let's take a look at it. It calculates (x-h)*s mod q. Looks legit? Not! If x<h then x-h is negativ! That's not going to work. (x-h)*s will be negativ too (remember: s is positiv) and the mod q reduction will not return the desired value. We have to take care of the case x<h. I think the easiest way is to reduce the x and h mod q in the beginning. Only then you can test for a negativ result by looking at the highest bit and, in case it is set, add q to the result making it positive (If you don't reduce x and h mod q then you can't use mula_small because you can't test the result by looking at the highest bit). So the new method Curve25519.sign() should look like this: private static final void reduce(byte[] x) { byte[] tmp=new byte[32]; divmod(tmp, x, 32, ORDER, 32); if ((x[31] & 0x80) != 0) { // x is negativ, add q to it mula_small(x, x , 0, ORDER, 32, 1); } }

public static final boolean sign(byte[] v, byte[] h, byte[] x, byte[] s) { // v = (x - h) s mod q int w, i; byte[] h1 = new byte[32], x1 = new byte[32]; byte[] tmp1 = new byte[64]; byte[] tmp2 = new byte[64];

// Don't clobber the arguments, be nice! cpy32(h1, h); cpy32(x1, x);

// Reduce modulo group order reduce(h1); reduce(x1);

// v = x1 - h1 // If v is negative, add the group order to it to become positiv. mula_small(v, x1, 0, h1, 32, -1); if ((v[31] & 0x80) != 0) { mula_small(v, v , 0, ORDER, 32, 1); }

// tmp1 = (x-h)*s mod q mula32(tmp1, v, s, 32, 1); divmod(tmp2, tmp1, 64, ORDER, 32); for (w = 0, i = 0; i < 32; i++) w |= v = *tmp1; return w != 0; }*

Answer from Come-from-Beyond: https://bitcointalk.org/index.ph ... g4645852#msg4645852 Thank u for the analysis. This bug was required to injected the serious flaw. In one of next versions we'll fix it.

1.8 Nxt 特性

1.8.1 别名系统

-与 DNS 类似

1.8.2 任意信息

- 任何人都可以发送任何形式的信息

1.8.3 资产交易

- 货币/股票交易

1.8.4 分布式计算

1.8.5 分布式存储

1.8.6 瞬时交易

1.8.7 混合服务

1.8.8 多重签名

1.8.9 服务供应商

- 区块链之外的服务

1.8.10 缩减

- 缩减膨胀的区块链

1.8.11 智能合约

智能合约的期望是将合约嵌入到有价值的且以电子方式控制的资产中[17]。Nxt 的只能合约可以用于发行分布式自治组织 DAC。DAC 可以作为 Nxt 矿池。

1.8.12 双相支付

1.8.13 投票系统

对于 Nxt 的可扩展性而言最重要的就是可以增加新特征。新特征使得 Nxt 更具有活力并且会吸引较大的用户群。同时也备受期待的是，添加的特征越多，就会产生越多的交易费---就增加了锻造的积极性，因此也就增强了网络的安全性。为了达到这个目的， Nxt 建立了一个投票系统，它可以让整个社区投票达成共识来确定哪个特性应该被实施以及什么样的顺序实施。但是投票系统本身并不是严格的技术创新，因为这是每个区域都可以实施的---但是 Nxt 已经将其植入系统了，希望能很快成为现实。

任何人都可以根据自己的需要发起投票。发起投票的人需要确定投票内容和投票期限（与区块数目相关联）。用户可以用它来解决所有的问题，比如选择新的图标。添加新的特征必须要通过 Nxt 股东们的投票决定允许。股东们也能投票决定别名转移和 Nxt 的更小单位。也可以投票决定毁灭（冻结）特定的钱币，特别是小偷或黑客们的钱币。甚至可以通过明主投票来决定停止恶意攻击的节点。更进一步说，社区可以通过投票来决定是否需要发起投票来考虑个别用户或节点。

投票是根据所持有的 Nxt 的数量来进行计算的。拥有较大交易账户的用户在这个系统里有较大的投票能力。为了防止这种情况，网络需要有健康数目的交易账户供用户选选择。另外，随着去中心化交易的实施，用户们也可以选择避免中心化交易。有了去中心化交易系统，就不存在中心化交易投票能力的问题了。

Nxt 的投票系统是去中心化货币的重要组成部分之一。这里没有领导者，没有集权的实体，所有的决定都取决于民主化投票。另外，除了能够解决全球问题，投票系统还能够被资产股东用于资产交易功能。它能帮助资产股东达成共识。

1.9 BCNext 的引述

讨论 Nxt 的区块产生是一个很好的建议。

https://bitcointalk.org/index.php?action=profile;u=152600;sa=showPosts;start=201

爱丽丝的账户上有 2500 个 nxt。上一次的四天前，她建立了一个区块。她的钱就像是矿机一样，拥有分散的权利将近 2500*4=10000GH/s。鲍勃的账户上有 1000 个 nxt，他去度假了而且 20 天都没有打开过他的账户。他的钱就像是矿机 1000*20=20000GH/s 的分散能力。爱丽丝和鲍勃每打开他们的账户一次，他们都有机会生成一个区块。他们拥有的钱越多，生成的机会也就越多。过了很久之后，源区块比生成区块的机会都要多很多。甚至是一个很小数额的账户都能意外的生成一个区块。持有币的顾客所产生的一个交易能够提供给服务器 0%的资金。始终通过区块产生来赚钱。

即时意味着不需要等待

https://bitcointalk.org/index.php?topic=316104.msg3392998#msg3392998

即时交易会在信息特征后得以实现。它的运行方式如下：

1.你在区块链上发布一条信息，宣称你的账户是特殊账户。

2.你用这个账户进行即时交易。

3.在某些时候，你需要清空你的账户，所以你发布了一条信息说你的账户应经变成了普通账户。

4.现在你就不能进行即时交易了，而且要一直等到 1440 个区块的时候（约 24 小时）才能去提取所有的币。

你可以不计其数的发送十分之一的余额，但除了其中的一个之外，所有的交易不会包含在这个区块里。节点看见只有一个交易的概率是可以忽略不计的，如果商家注意到其它的交易超过了一天的限制，他们不会把商品卖给你。

255 的交易量是一个好数字。255 与 8bits 相符合，而且 255*128b 与 32kb 相符合。

https://bitcointalk.org/index.php?topic=303898.msg3347084#msg3347084

更新

添加一个按钮来生成可用于第三方网站的授权标示。这些标示可以用来识别账户的拥有者，从而减少注册的必要性。通过标示，任何网站都能得到账户 ID（用于提款和其它需要），并确保用户就是该账户的合法拥有者。

授权工作流程

1】在 Nxt 客户中解锁一个账户

2】按下 KEY 按钮并输入网站网址

3】复制生成的标示（160 个字符）并粘贴到网站进行登录

保持更新，技术细节：

https://bitcointalk.org/index.php?topic=303898.msg3298139#msg3298139

地址的长度：8B（在人类可读的范围里最大 20 个字符）

私钥地址长度：32B

公钥长度：32B

签名：64B

区块标头长度：192B

交易长度：128B

每个交易都有“截止”期限来限制时间，以保证期能加入到一个区块之中。当你发送了一个交易时，你再也不用等待一天来确认了。现在你可以将截止日期设置到 5 分钟，之后如果没有确认你可以重新发送，还可以选择更高的费用。同样的，每个交易都有 ID(长度 8B)以帮助商家来识别你的支付。不需要为每个顾客重新设置一个单独的地址了。

更多细节

https://bitcointalk.org/index.php?topic=303898.msg3361014#msg3361014

每个区块都有“产生签名(generationSignature)”参数。一个激活账户用自己的密匙来签署之前区块的产生签名。这是通过 SHA256 哈希散列而产生的 64 个字节。Hash 的前八个字节产生了一个数字（把它称之为"hit”）。hit 与现有的“目标”（64bit 的数字）相对比较。如果 hit 比目标值要低，下一个区块就可以生成了。

每个账户的目标值与其余额是成比例的。持有 1000 个币的账户得到的目标值比持有 20 个币的账户得到的目标值要大 50 倍。因此拥有 1000 个币的账户锻造的区块数是持有 20 个币的账户锻造的区块数的 50 倍（从长远运行来说）
目标值并不固定的，因为之前区块的时间在变化，因此它每秒钟都在增大。如果在第一秒没有人能生成区块那么目标值就会变成之前的两倍，以此类推。基本目标值是 60 秒标记处的目标值。如果只有很少数目的激活账户，那么长时间之后就会有人生成一个区块，因为目标值会变得很大。如果你打开了客户端并登陆，你会在 BLOCKS 部件处看到计时器。它就会告诉你什么时候这个目标值能够比你的 hit 值要大。

Nxt 开发的时候考虑到了交易隐私
https://bitcointalk.org/index.php?topic=303898.msg3355477#msg3355477
我还没有泄露任何细节，但是在 Nxt 里面有一个特殊的账户(recipient #0)，它将会被用于高级特征（混合服务，第三方担保，等等）。

https://bitcointalk.org/index.php?topic=303898.msg3372610#msg3372610
假设你持有 10%的股份，和一台从 51 区偷来的外星人电脑。
以一分钟一个区块的速率来生成 60 个区块长链的几率是很小的(就像 1/10000000000000...)。你的分支累计难度比主链难度小很多，因此不会被其它节点所接收（你可以很快的的产生你的区块链）。你持续寻找私钥，而这个私钥能产生 hit 值接近于 0 的公钥（称之为攻击账户）。你需要做很多的 Ed25519 计算（比 SHA256 所需要的 CPU 能力很多），但是你有外星人电脑，因此可以在 2 分钟国内很快的完成这些任务。现在你必须要再去 51 区偷一台时光机或者等待几个月（否则你的区块链将会因为不正确的时间戳而不被接受）。

如何对付攻击
第一个进入我脑海的想法是将成熟期从 60 个区块（1 小时）延长到 1440 个区块（1 天）。这迫使你去等待数年，甚至数年之后由于检查点，你都没法完成攻击。我不喜欢在我产生一个区块之前等待一整天。我会想很多其它的解决办法。

https://bitcointalk.org/index.php?topic=303898.msg3393070#msg3393070
有 2^256 个可能的地址，剩余的 192bits 目前并没有应用。未来我们可能会使用剩余的 64bit 来拓展地址的可视部分。然后，所有目前现存的 20 个字符长度的地址会变成 40 个字符。现在，如果有人能找到一个地址拥有相同的 64bits，它们将不能再发送交易了。

Nxt 需要十倍小的带宽以及储存量，它使用的签名算法比 Bitcoin 要快很多。因此它能够达到比比特币更高的规模。目前还没有一个已经实施了的针对 VISA 这么大交易量的去中心化的解决方案。我们难道不应该做点什么吗？

首先，没有人能进行双重支付，节点不会产生这样的区块，也不会接受它们
https://bitcointalk.org/index.php?topic=316104.msg3397005#msg3397005
十分之一的规则目的是为了在一下情景中保护商家们，即他们没有看见能够确认的交易而导致取消商家已经接受的交易。在尝试双重支付之前，需要发送至少 10 个交易来实现这个发送目的。如果商家们看到了至少其中的一个交易，他们就会拒绝出售商品。10 个交易都不能到达商家的概率是很小的，可以忽略不计。网络通过拒绝确认违反了 1/10 规则的交易来帮助保护商家。如

果用户发送了两个交易（第一个给自己，第二个给商家），如果第一个交易被确认了，那么商家需要等待稍长的时间来收到钱。这看起来违背了规则，但其实是完全不重要的。

去中心化的应用商店

https://bitcointalk.org/index.php?topic=317607.msg3405569#msg3405569

Nxt 拥有很多特征来实现去中心化，其中一个特征是信息发送，它可以纯文本来买卖虚拟产品。特别是用来出售计算机或手机应用。

工作流程很简单：

1】开发者发送信息来描述产品（存储在区块链中）。

2】用户发送支付到开发者账户。

3】开发者发送加密信息，可以是链接或者代码（也存储在区块链中）。

4】用户解码加密信息，并使用代码来解锁应用。

关键点：

1】开发者需要将他们的账号公布在网页上以证明他们是合法的拥有者（或者使用 Nxt 的信用系统）。

2】所有的操作都是公开的，且可以被审核的（如果其中一方披露密码，就可以审核加密信息）。

3】除了交易费，其它的利润都 100%给了开发者。

4】没有人能够关闭商店或者 DDOS 商店。

Nxt 客户端将会用一些小工具来扫描区块链和可视化虚拟产品。我想要看到客户端软件会实施什么样的特征。

https://bitcointalk.org/index.php?topic=303898.msg3411024#msg3411024

区块的产生是由之前的区块决定的。如果你在 3 小时之内能产生下一个区块，你可以立即准备一个区块，但是在能披露它之前你需要等待 3 个小时（否则就会被网络忽略）。你可以同时建立很多个区块来找到最适合攻击的一个，但是，在某些时候你需要其他人来生成区块（或者再次等待）。Nxt 使用了全新的 POS 算法，并非像 PPC,NVC 和其它密码学货币采用的算法。Nxt 没有漏洞。

Nxt 支票

https://bitcointalk.org/index.php?topic=322935.0

我们计划创建不同面值的 Nxt 支票。每一张钞票将包含如下参数：

1.宽度/高度：大小和信用卡一样，圆角（我们讨论选择像美元一样的尺寸大小，但是 Nxt 更国际化）。

2.密度：220gsm（密度是办公用纸的 3 倍）

3.背景颜色：蓝灰色（RGB-171A1D）

4.主要的文字颜色：象牙色（可以用金色，但是不要用来印刷在蓝灰色上）

5.保护：信息图（10 种面值，1,2,5,10,20,50,100,200,500,1000），信息图下面有可兑换的代码。这些支票可以同纸币一样使用，但是它们的主要目的是吸引普通人。用户可以通过在客户端输入代码来兑换支票。

我们也面临着一些问题：

1.订单最少为 100,000 张支票（ 1,000,000 可以打折）

2.在获得全息图的时候部分代码可能会被移除，因此我们需要选择其他的背景色，但是蓝色看起来很酷。
3.粘贴全息图会使得成本增加，还要去想一些其他的方法来进行保护

增加链交易的保护

链交易是什么。每个交易可以参考其他的交易。举个例子来说，一个信息型的交易可能会参考支付型的交易来描述支付会用于什么。如果一个交易参考了其它的交易，那么只有在被参考的交易得到确认后，该交易才能被添加到区块中。
可以用不同的方法来使用链交易。SatoshiDice--类似的服务能够接受未确认的交易，并进行即时交易。唯一需要的就是参考一个赌注交易。如果赌注没有被确认（由于双重支付，等等），则奖励交易也不会被确认。你需要思考的就是链交易可用来做什么。

https://bitcointalk.org/index.php?topic=303898.msg3472657#msg3472657
此外，除了去中心化的 DNS 外，Nxt 还有去中心化的文件储存。Nxt 的客户端将会架设在区块链上，从我的 Nxt 账户上会发布相应的更新（跟产生源区块一样）。我希望我的 NXt 股份足够促使所有的更新。目前版本所占据的容量大小与 1280 个支付交易是一样的，存储 160KiB 需要支付的费用是很昂贵的。

https://bitcointalk.org/index.php?topic=303898.msg3507997#msg3507997
Nxt 的 DNS 是否真的与 Namcoin 的 DNS 不同是有争议的。但是 Nxt 的有一个很大的优点——能够从盒子里开始工作。而 Namecoin 的开发者在讨论/计划/想象我写的代码。我不认为 Namecoin 的 DNS 是一个竞争者，一旦 Nxt 的 DNS 发布，NMC 就会变成一个山寨币。

文件储存可用于任何文件。让区块链膨胀的因素是没有区别的——不管是支付交易还是文件。如果能够支付费用那是很好的。每年的修剪将会去除掉所有的文件，因此需要重新上传它们。目前区块链被限制在每年 16GiB，但是平均下载 8GiB 并不是个大问题。

最低等级的就是信息（包括合约，文件存储，DNS）、彩色币、投票系统、信用系统。高级（用户友好型）的是图书馆、商店（类似去中心化的应用商店）、资产交易，等等。

额外的功能可以通过向服务提供商支付 Nxt 来添加。Mini-blockchains 非常适合于服务提供，另外，大型的文件也可以用 SP 来存储。
在继续这份声明的时候我们都在寻找更多的 Nxt 核心开发者，以下特征是我们计划在区块高度 52596 后开始实施的（按照字母顺序）：

1.10 Come-from-Beyond

https://bitcointalk.org/index.php?topic=345619.msg4172293#msg4172293

- 别名系统（已完成）——与 DNS 相似
- 任意信息（部分）——任何软件都可以发送任何形式的信息
- 资产交易（已完成）——货币/股票交易
- 分散的计算（概念阶段）——与 BOINC 相似

- 分散的储存——与 Torrent 相似
- 即时交易（需要节点审查）——保证确认的交易
- 混合服务（概念阶段）——洗钱服务
- 多重签名——多重签名
- 服务供应商——链条之外的服务
- 缩减——膨胀区块链的缩减
- 智能合约（概念阶段）—— http://en.wikipedia.org/wiki/Smart_contract
- 透明锻造（部分）——无法用几句话来描述
- 双相支付——需要额外确认的支付
- 投票系统——民主

透明锻造：

透明锻造会包括导致锻造账户“集团化”的一些变化。最主要的目的是为了无法预测谁会锻造下一个区块。如果 Alice, Bob, Charlie 和 Dan 是锻造区块的候选人，那么 Alice, Bob 或是 Charlie 都能进行，不管 Alice 是否能早 5 秒比 Bob 和 Charlie 锻造出。区块之间准确的间隔为 60 秒钟，而时间戳只是用于决定事件顺序。如果我们已经知道谁能锻造出下一个区块，我们就没必要再等 93 秒钟。我称之为“时间错位”。如果 Bob 的区块赢得了这场比赛，那么 Alice 和 Charlie 就被惩罚 1440 个区块。

1.11 结论

只有时间能够给出答案，是否 Nxt 将会真正引起巨大的电子经济变革，它们将替代全球 GDP 很大的比例。Nxt 也确实拥有所有的特征来允许其位于第一位。比特币照亮了这条道路。Nxt 紧随其后并更加夺目。

1.12 参考

1.https://bitcointalk.org/index.php?topic=345619.msg4383169#msg4383169
2.http://blockchain.info/charts
3.Washington Post 12 Nov 2013[size=12.666666984558105px]-[size=12.666666984558105px]http://www.washingtonpost.com/blogs/the-switch/wp/2013/11/12/bitcoin-needs-to-scale-by-a-factor-of-1000-to-compete-with-visa-heres-how-to-do-it/
4.[url=https://blockchain.info/charts/difficulty]https://blockchain.info/charts/difficulty[/url]
5.https://blockchain.info/charts/hash-rate
6.[url=http://money.cnn.com/gallery/tec ... oin-mine/index.html]http://money.cnn.com/gallery/technology/2013/12/17/bitcoin-mine/index.html[/url]
7.https://blockchain.info/pools
8.http://hackingdistributed.com/2013/11/04/bitcoin-is-broken/
9.http://www.coindesk.com/bitcoin-miners-ditch-ghash-io-pool-51-attack/
10.https://blockchain.info/pools

11.http://www.forbes.com/sites/reuvencohen/2013/11/28/global-bitcoin-computing-power-now-256-times-faster-than-top-500-supercomputers-combined
12.http://blockchain.info/stats
13.http://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange
14.Yung, M., Dodis, Y., Kiayias, A., Malkin, T., & Bernstein, D. J. (2006). Curve25519: New Diffie-Hellman Speed Records. In , Public Key Cryptography - PKC 2006 (p. 207).
15.http://grouper.ieee.org/groups/1363/P1363a/contributions/kcdsa1363.pdf
16.http://qz.com/165273/the-existential-threat-to-bitcoin-its-boosters-said-was-impossible-is-now-at-hand/
17.http://szabo.best.vwh.net/smart_contracts_idea.html

英文版白皮书：http://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt