Filecoin：一个以加密货币运转的文件存储网络

1e96a1b27a6cb85df68d728cf3695b0c46dbd44d
filecoin.io

July 15, 2014

抽象

Filecoin 是一种和比特币类似的分布式电子货币。而和比特币只用算力的工作量证明不同，Filecoin 的工作证明函数还包含了可获取性证明部分，要求节点证明自己存储了指定的文件。Filecoin 网络形成了一个完全分布式的文件存储系统，鼓励节点尽可能地存储整个网络的数据。存储文件能收到货币奖励，而且和比特币一样，货币能在交易中流通。花钱能把文件添加到网络里。这就产生了强大的货币刺激，促使独立个体来加入到网络里工作。在 Filecoin 网络日常操作的过程中，节点贡献出来的有用工作是对有价值数据的存储和分发。

1 介绍

许多计算机系统的数据存储和访问都是服务商提供的。现在少数的几个大型服务提供商提供了这种市场上的大部分服务。很少有新的市场参与者，因为与现有服务商直接进行全面地竞争几乎是不可能的。这导致了某些低效的情况很难被优化。比如，今天数据传输的瓶颈几乎就是最后一英里的 ISP(译者：而这恰恰就是中心化的服务提供商所无法解决的网络传输线路问题)，骨干网络和本地局域网都远比 ISP 快。如果让代理们能拥有个人动机去存储数据和优化本地分发，这种基于分布式服务的解决方案就能带来一个巨大的进步。

另外，分布式存储系统一个主要的目标是确保重要文件的保藏。现在的存储系统在这方面很脆弱。首先，它们倾向于让一个服务提供商集中管理文件，这就把文件的命运和负责存

储的提供商的命运绑在一起了。第二，在比如 HTTP 这样被广泛使用的文件检索体系里，一个文件是被其位置标识的，而不是被其内容。这导致文件的可用性依赖于特定主机的正常运行时间，而不是存在于任何网络位置上的文件本身。第三，目前广泛使用的体系都把文件服务的永久负担放在了原始创建者身上 — 要么自己负责托管文件，要么雇用服务商来做 — 这通常都是不可持续的，特别是对于科学数据集一类的大型文件。所以现在真正需要的是一个全球化的分布式网络，其上的独立代理能为任何文件服务，并有很强的激励机制来刺激它们提供服务。

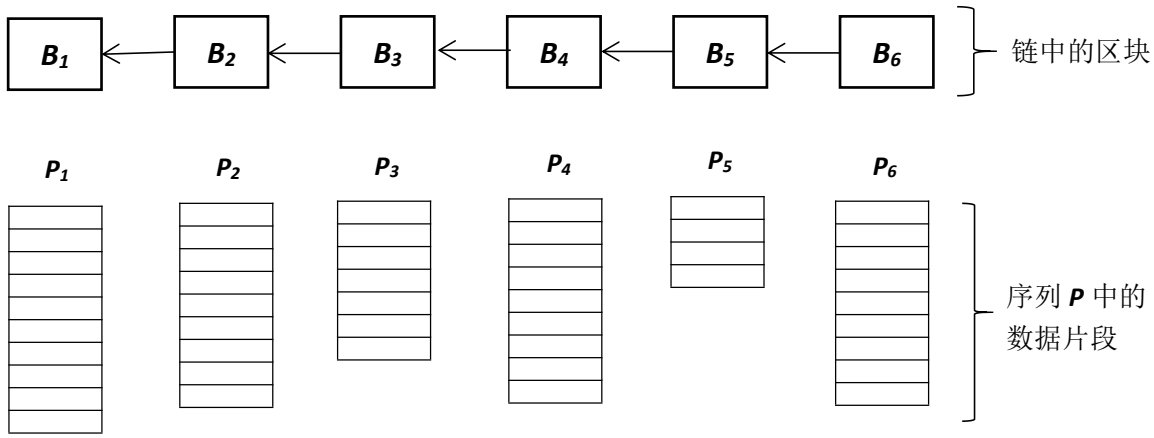
最近发展出来的基于区块链的加密货币，能组织并刺激出一个庞大的机器网络，为其做一些如工作量证明的计算。在比特币[7]里这些计算几乎是无用的。已经有其它货币尝试了更有用的算法工作，比如 Primecoin[4]会奖励在素数链上的新发现。其实这些系统也可以为一些清晰紧急的需求执行其它种类的有用工作，比如文件的存储和分发。在本文中，我们提出了一个以类似比特币的加密货币网络来刺激文件存储的解决方案，它的工作函数纳入了可获取性证明。

2 设计

Filecoin 和比特币一样通过区块链实现了交易账本，每个区块的诞生都必须伴随着基于加密哈希函数的工作量证明。工作量证明的难度可以通过参数动态调整，Filecoin 和比特币一样会把区块的出现频率大致控制为十分钟一个。还有一点和大多数基于区块链的系统一样的是，当一笔交易所在的区块建立之后，又建立了几个有效区块地时候，客户端才应该认为该交易被确认了。这几个有效区块的数量取决于，为限制敌对节点的能力而做的假设(比如，防止敌对节点能控制全网算力的大部分)。虽然有这么多类似，Filecoin 在标准比特币风格的设计上还是做了一些核心的改变。

2.1 片段集合

Filecoin 的第一个关键组件是新添加的数据*片段*[1]增长序列 ,片段能组成被网络存储的文件。一个*片段*是数据的一个不透明分片，其大小固定为 **S**，是一个可调节的网络参数[2]。Filecoin 区块链通过一个特殊的 **Put** 交易(2.2 节)，把新添加的片段放入到某个区块 $\mathbf{B_t}$ 中。*片段集合* **P** 按时间先后排序。和比特币一样，所有节点必须在本地存储整个区块链，但片段分布在整个 Filecoin 的节点之间。这个方案让 Filecoin 区块链能提供数量级远大于区块链本身的数据存储能力。



$$P = P_1 \,||\, P_2 \,||\, \ldots \,||\, P_t$$

[1] 文件暗示了任意大小的数据分片，而 Filecoin 使用了固定大小的数据分片。为了避免混淆，*block* 指形成了 Filecoin 区块链的区块，*piece* 指被区块链存储的一块数据。

[2] 每个片段（piece）都给被全网存储的区块链带来了开销。**S** 的选择还取决于可获取性证明方案里的参数(2.5 节)。

2.2 Put 和 Get 交易

Filecoin 第二个关键添加是 **Put** 和 **Get** 交易规范。**Put** 交易把文件添加到网络存储里：每个 **Put** 交易包含一个新添加片段的记录列表。一个*片段记录*是一个如下格式的值：

$$\text{record}_i = (H(p_i), H(\sigma_i), pk_i, reward_i)$$

$\mathbf{p}_i$ 是一个片段，$\mathbf{H}(\mathbf{p}_i)$是其加密哈希，$\boldsymbol{\sigma}_i$ 是 $\mathbf{p}_i$ 的一组认证(2.5 节)，$\mathbf{pk}_i$ 是一个验证公钥(2.5 节)，$\mathbf{reward}_i$ 是一些奖励参数(2.7 节)。当一个 **Put** 交易被确定后 - 做为被挖区块的一部分 - 该区块所有片段记录所标识的片段就可以被认为已被网络存储。

一个 **Get** 交易用于展现一个指定的，在以前按要求被存储的片段。当一个 **Get** 交易被确定后，它所列出的片段会被传输给交易发出者，如在 2.4 节中所述。这两种交易以标准的方式扩展了比特币协议，用于存储和获取能由串联的片段组成的文件。

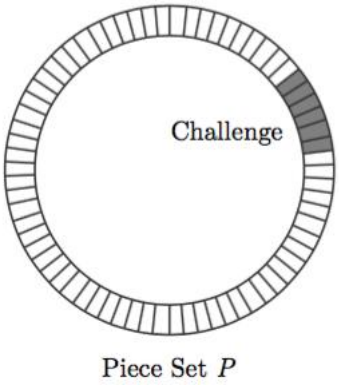
2.3 通过挑战刺激节点存储片段

Get 交易还不足以刺激矿工存储所有的片段，因为矿工不能预测未来 **Get** 交易的分布，以及自己的预期回报。Filecoin 与比特币风格的加密货币相比，最重要的变化是对工作函数的修改。为了成功挖到区块，矿工在基于哈希的工作量证明之上，还要完成一组挑战，以证明自己现在存储了一组特定的片段。

比如，假设存储了片段记录的区块链最新区块是 $\mathbf{B}_{t-1}$，为了把区块 $\mathbf{B}_t$ 添加到区块链里，矿工必须证明自己存储了一组指定的片段，它们由挖掘中的区块 $\mathbf{B}_t'$ 决定($\mathbf{B}_t'$ 中包含了一个比特币风格的工作量证明)，在 2.6 节有述。具体点说，起始片段索引 i_t 的决定方式如下：

$$i_t = H(B_t') \bmod |P|$$

索引指向 **P** 中的一个子片段序列 $\mathbf{p}_i$，i 的范围是从 i_t 到(i_t + k − 1)，k 是一个可调节的难度参数。索引均匀地选择了样例片段。挑战次数 k 灵活地控制了挑战的难易程度，以匹配节点的能力大小。比如，k = 6 会要求节点证明自己拥有了片段 i 到 i + 5。我们发布连续片段的挑战，是因为非连续片段集合可能会在存储上产生超线性的预期，从而刺激了大型矿池的出现。和比特币一样，这样的矿池可能会让全网面临被敌对节点控制大部分能力的威胁。



如果矿工想通过挖掘区块 $\mathbf{B}_t$ 来延长区块链，并以此领取奖励，就必须提供一个证据以表明自己当前正存储着这组挑战片段 $\mathbf{p}_i$：

$$PieceProof(p_i, t) = (H(B_t^{'} \,||\, p_i), \pi_i)$$

$\mathbf{B}_t^{'}$ 是目前建造出的，且正处于挖掘中的区块，要由挑战确定，而值($\mathbf{H}(\mathbf{B}_t^{'} \,||\mathbf{p}_i)$, $\boldsymbol{\pi}_i$)则构成了矿工正存储着挑战片段的证据(2.5 节)。由于全网的存储 - **P** 中的片段 - 快速的增长超出了独立节点的存储能力，所以要刺激矿工去获取那些被最少的其它节点存储地片段，因为这样能有一个明显的机会能在未来在挖矿挑战上产生收益。

如 Filecoin 的系统必须考虑到某个片段所有的副本都丢失的问题。这会导致某些挑战无法被完成，也有可能引起大的网络失败问题和数据丢失问题。不过到现在谈论过的结构中，这还不是一个问题，因为确认区块所需的挑战，依赖于一个确认中的含有工作量证明的区块 $\mathbf{B}_t^{'}$。两个节点发现了两个有效但不同的工作量证明就会面临不同的片段挑战。

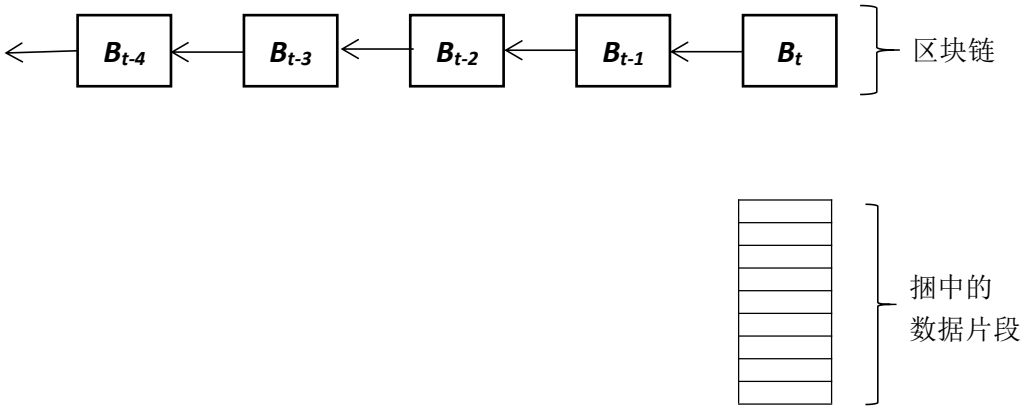
2.4 分散捆里的片段

如前文所述，Filecoin 激励节点收集片段，激励节点为自己保留最稀有的片段，从而提高自己预期的挖矿奖励。Filecoin 提供了一个片段分散机制，以防止垄断和整体囤积。当挖到新块时，节点还必须出示一个片段集合，叫做*捆(bundle)*，里面包含了区块链新的头部区块所引用的所有片段 $\mathbf{p}_i$(和对应的$\boldsymbol{\sigma}_i$)。这里面有挑战片段、**Get** 和 **Put** 交易里的片段。捆这种方

式一次性地解决了多个问题：

- 它不鼓励囤积，防止节点利用长久片段来垄断。
- 它确保了片段被发送给 **Get** 交易的发出者。
- 它确保了 **Put** 交易的新片段能被很多节点都看到，这样这些片段就能快速地被多个节点存储，从而在冗余存储上得益。
- 它为新加入网络的矿工提供了起始片段集合。
- 它降低了伪造可获取性证明的可能性，在 2.5 节有讨论。

一个捆内片段的数量大约为数百个交易的数量，这些交易只是区块链非常小的一个部分(当前比特币网络的区块链超过了 15G)。



2.5 以压缩版的可获取性证明来验证

Filecoin 区块链必须能被网络上的任意节点验证，包括没有先前数据的新节点。Filecoin 可获取性证明的证据 **PieceProof**($\mathbf{p}_i$,t)有两个组件：$\mathbf{H}(\mathbf{B}_t' \| \mathbf{p}_i)$和$\boldsymbol{\pi}_i$。第一个组件，哈希值$\mathbf{H}(\mathbf{B}_t' \| \mathbf{p}_i)$，允许任意有原始片段 $\mathbf{p}_i$ 的节点能验证区块 $\mathbf{B}_t$ 的矿工在挖掘时拥有 $\mathbf{p}_i$。第二个组件$\boldsymbol{\pi}_i$，是一个压缩版的可获取性证明，后面对它做了定义，它能被任意节点验证，即使没有原始片段 $\boldsymbol{p}_i$。

由于基于哈希值 $\mathbf{H}(\mathbf{B}_t' \| \mathbf{p}_i)$的验证需要访问原始片段 $\mathbf{p}_i$，所有以这种方式验证整个区块

链是非常昂贵的。取而代之的是，Filecoin 节点只在挖掘最新区块时，才进行这种昂贵的验证(如 2.4 节描述，相应的片段会以“捆”的形式伴随每个新块传播)，并依赖压缩版可获取性证明$\boldsymbol{\pi}_i$验证区块链上更早的区块。

我们现在来更详细地描述可获取性证明的使用。大多数可获取性证明涉及到两方，一个*客户端*预处理数据，一个*服务端*存储被处理的数据。在任何时候客户端都可以向服务端发起*挑战*，服务端必须为挑战计算出相应的*证明*。通常，挑战的生成和验证都使用了一个只有客户端知道的密钥。然而，在不久前的方案里，比如 Shacham 和 Waters 的[8]以及 Ateniese[1]等人的方案，期望了如下属性：

1．可公开验证：挑战是任何人都可以发起和验证的，不只是原客户端。

2．带宽占用小：挑战和证明都要足够小，小到可以被添加到区块链里。

为了能把这些方案用在区块链里，我们调整了这些方案，通过把公钥存储在片段记录里，然后计算挖掘中区块的哈希，以此发布挑战，最后把可获取性证明存储到生成的区块头里。

$$(\mathbf{sk}_i, \mathbf{pk}_i, \boldsymbol{\sigma}_i) \leftarrow \mathbf{PoR.Setup}(\mathbf{p}_i)$$

$$\mathbf{challenge} \leftarrow \mathbf{PoR.Challenge}(\mathbf{pk}_i, \mathbf{H}(\mathbf{B}_t' \,||\, i))$$

$$\boldsymbol{\pi}_i \leftarrow \mathbf{PoR.Prove}(\mathbf{challenge}, \mathbf{p}_i, \boldsymbol{\sigma}_i)$$

$$\mathbf{PoR.Verify}(\mathbf{pk}_i, \boldsymbol{\pi}_i) \in \{0, 1\}$$

$$\mathbf{record}_i = (\mathbf{H}(\mathbf{p}_i), \mathbf{H}(\boldsymbol{\sigma}_i), \mathbf{pk}_i, \mathbf{reward}_i)$$

$$\mathbf{PieceProof}(\mathbf{p}_i, t) = (\mathbf{H}(\mathbf{B}_t' \,||\, \mathbf{p}_i), \boldsymbol{\pi}_i)$$

PoR 可以通过 Shacham 和 Waters[8]的公开验证方案初始化。对于预期新区块里的每个片段 $\mathbf{p}_i$，矿工都会为其创建一个密钥对($\mathbf{sk}_i$, $\mathbf{pk}_i$)。我们把 $\mathbf{H}(\boldsymbol{\sigma}_i)$和 $\mathbf{pk}_i$ 存储到 **Put** 交易的片段记录里。在验证 **Put** 交易期间，也必须验证认证$\boldsymbol{\sigma}_i$，这在 2.6 节讨论。任何节点要证明

自己拥有片段 $\mathbf{p}_i$，就必须存储 $\mathbf{p}_i$ 和它的认证$\boldsymbol{\sigma}_i$。挑战片段是由区块决定的，$\mathbf{H}(\mathbf{B}_t' \| i)$也决定了挑战的随机性。最终被确认的区块上回存储挑战的证明。这就为整个区块链的验证提供了一个压缩的，可公开验证的方式。然而，这个对 Shacham 和 Waters 的公开验证方案[8]的调整，在我们的上下文里有个弊处：如果攻击者成功挖到过区块，那么他就能保留这个区块的私钥 $\mathbf{sk}_i$，以后就能在未处理原始片段数据 $\mathbf{p}_i$ 和$\boldsymbol{\sigma}_i$ 的情况下，用这个私钥伪造一个有效的可获取性证明$\boldsymbol{\pi}_i$。为了防止这种伪造，我们要求新挖的区块要有哈希值 $\mathbf{H}(\mathbf{B}_t' \| \mathbf{p}_i)$，并要求矿工分发原始片段数据($\mathbf{p}_i$, $\boldsymbol{\sigma}_i$)(2.4 节)，其它节点检查输出的哈希值 $\mathbf{H}(\mathbf{B}_t' \| \mathbf{p}_i)$。有了这种防御措施，Filecoin 在某些情况下就能提供比比特币更强的安全保障：尤其是在一个拥有全网 51%哈希运算能力的对手存在时，即使它生成且存储了 $\mathbf{sk}_i$，并因此能伪造相应的可获取性证明，也还是必须要在伪造的时候花费资源去存储或获得数据片段。

2.6 区块的构造和验证过程

在挖掘一个新的 Filecoin 区块 $\mathbf{B}_t$ 时，需要准备包含的交易、构造区块并准备相应的数据**捆**。

- **交易**被组装成标准的默克尔树结构，根为 **txRoot**。
  - **Put** 交易包含一组新片段的记录：$\mathbf{record}_i = (\mathbf{H}(\mathbf{p}_i), \mathbf{H}(\boldsymbol{\sigma}_i), \mathbf{pk}_i, \mathbf{reward}_i)$。
  - **Get** 交易包含了片段 i 的哈希值 :$\mathbf{H}(\mathbf{p}_i)$,以及 2.3 节描述的值 :**PieceProof**($\mathbf{p}_i$, t)。
  - 其它类型的交易，和比特币一样。
- **区块** $\mathbf{B}_t$ = ($\mathbf{parent}_t$, $\mathbf{txRoot}_t$, $\mathbf{nonce}_t$, $\mathbf{PoW}_t$, $\mathbf{PoR}_t$)，这里：
  - $\mathbf{parent}_t$ 是 $\mathbf{H}(\mathbf{B}_{t-1})$。
  - $\mathbf{txRoot}_t$ 是交易的默克尔树根。
  - $\mathbf{nonce}_t$ 是临时选中的，这和比特币的工作量证明方案一样，所以：

$$\mathbf{PoW}_t = \mathbf{H}(\mathbf{parent}_t \| \mathbf{txRoot}_t \| \mathbf{nonce}_t) < 2^l$$

其中参数 l 用于动态调整哈希运算的难度。

- $\mathbf{PoW}_t$ 是工作量证明的输出值，由其决定了挑战片段的索引值，如在 2.3 节所述
- $\mathbf{PoR}_t$ 是一系列 $\mathbf{PieceProof}(\mathbf{p}_i, t)$的值，也就是每个片段 $\mathbf{p}_i$ 的挑战证明。

- **捆**数据包含了当前区块挑战集的所有片段，以及区块内所有 **Put** 和 **Get** 交易引用的片段。

一旦区块 $\mathbf{B}_t$ 构造成功后，矿工就把它广播给网络里其它的节点。任何其它节点都可以通过以下过程验证区块 $\mathbf{B}_t$。

1．验证 $\mathbf{parent}_t$ 是否是之前的区块链头部区块。

2．验证所有的交易：

- 执行比特币风格的交易验证，包括检查余额的变化是否有效。
- 对于每个 **Get** 交易：
  - 检查 $\mathbf{p}_i$ 和$\boldsymbol{\sigma}_i$ 是否在捆中。
  - 检查 $\mathbf{H}(\mathbf{p}_i)$是否匹配。
  - 检查 2.5 节描述的 $\mathbf{PieceProof}(\mathbf{p}_i, t)$。
- 对于每个 **Put** 交易：
  - 检查 $\mathbf{p}_i$ 和$\boldsymbol{\sigma}_i$ 是否在捆中。
  - 检查 $\mathbf{H}(\mathbf{p}_i)$和 $\mathbf{H}(\boldsymbol{\sigma}_i)$是否匹配。
  - 检查 $\mathbf{reward}_i$ 是否小于交易发布者的余额。
  - 检查 $\mathbf{pk}_i$ 是符合 Shacham 和 Waters[8]里描述的组的一对元素。
  - 检查$\boldsymbol{\sigma}_i$ 是否构造的正确，要符合后面所述的过程。

3．验证 $\mathbf{PoW}_t = \mathbf{H}(\mathbf{parent}_t \,||\, \mathbf{txRoot}_t \,||\, \mathbf{nonce}_t)$，并且 $\mathbf{PoW}_t < 2^l$。

4．验证 $\mathbf{PoR}_t$ 里的 $\mathbf{PieceProof}(\mathbf{p}_i, t)$，如 2.5 节所述。

验证$\boldsymbol{\sigma}_i$是否构造正确了是件重要的事情，不然敌对节点就能提供无效的$\boldsymbol{\sigma}_i$了，这样有些时候其它矿工就不能构造有效的证明了。检查$\boldsymbol{\sigma}_i$会涉及到检查认证是否构造正确了：

$$e(\sigma_{i,j}, g) = e(H(j) \bullet u^{p_{i,j}}, v)$$

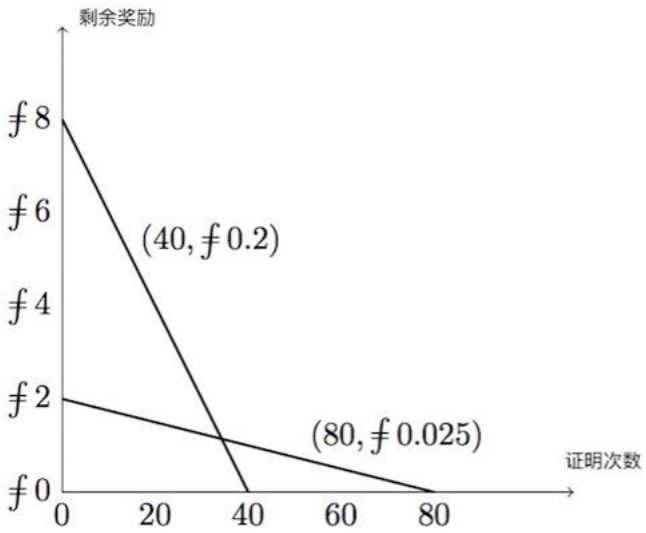
j 索引到片段 $\mathbf{p}_i$ 的每个子分片以及所有其如 Shacham 和 Waters[8]的解释中所描述的认证 $\boldsymbol{\sigma}_i$。为每个 **Put** 交易检查其所有$\boldsymbol{\sigma}_i$里的认证会带来显著的计算负担。而只随机验证一部分认证就有可能带来更高效的验证。我们注意到在这种情况下，所有的验证器必须通过哈希 $\mathbf{PoW}_t$ 来选择明确的认证子集，否则验证器之间可能就无法在接受或拒绝一个区块之间达成共识。

2.7 奖励参数和二级市场

Filecoin 允许在 **Put** 交易里指定片段的奖励。这种方式让用户能使用 Filecoin 货币来增强刺激特定片段的存储。每个片段的奖励参数 $\mathbf{reward}_i$ 必须在 **Put** 交易的 $\mathbf{record}_i$ 中指明。交易发起者必须先行支付所有奖励，把货币放入到有关片段的基金中。我们为奖励函数留下了充足的设计空间，所以这里只描述一种简单的方式，把更多其它的方式留给未来的设计。我们的函数使用了如下参数：

$$\text{reward}_i = (TTL_i, RPP_i)$$

$\mathbf{TTL}_i$ 是 $\mathbf{p}_i$ 的存储次数，$\mathbf{RPP}_i$ 是每份存储证据的奖励。我们的函数会额外奖励 $\mathbf{RPP}_i$ 数量的 Filecoin，给在存储挑战上成功的证明了自己拥有 $\mathbf{p}_i$ 的矿工，直到 $\mathbf{TTL}_i$ 次。奖励的 Filecoin 总数是($\mathbf{TTL}_i \cdot \mathbf{RPP}_i$)，**Put** 交易的发布用户必须先行支付。这两个简单的参数给了用户显著的自由度去控制刺激特定片段的存储。尽管要以“证明挑战”作为度量，**TTL** 还是相当于片段在网络里的一生，或者是刺激的时长。**RPP** 相当于刺激的强度。下图说明了($\mathbf{TTL}_i$, $\mathbf{RPP}_i$)两者之间配置的权衡：



在最简单的奖励参数设置中，会匀速发放片段 $\mathbf{p}_i$ 的挑战奖励（如上图所示），其所发放的挑战奖励相加之和，就是插入片段 $\mathbf{p}_i$ 的初始 **Put** 交易所用的货币数量。在其它的奖励参数设置中，可以把奖励的递减依赖于其它的，非线性函数（可能是指数递减），或者可以指定提取的总奖励超过 **Put** 交易插入的总奖励（从而创建可通货膨胀的货币）。注意，奖励方案的微小不同也可能导致截然不同的片段存储分布，或是可能完全破坏掉激励体系。修改奖励方案时必须仔细地分析其所能产生的市场平衡。比如，如果货币过于通货膨胀，那么攻击者就可能通过添加大量的“虚拟数据”来获益，他们可以容易的重复生产这些数据，而且不必负担存储的花费（比如一个攻击者知道秘钥的伪随机函数的输出），从而在长期的挑战奖励系统中获得净收益。

节点也可以从 Filecoin 网络中以非直接兑现挖矿奖励的其它方式获得收益。矿工可以设置额外的交易费，交易费取决于使用 Filecoin 网络做货币交易的客户端的总需求。另外，由于任何地方都可能存储和需要片段数据，并且，即使是有着超强哈希算力的矿工自己本身也不太可能存储所有产生的挑战片段，所以可能会出现片段数据的二级市场。在这些二级市场中，即使节点不能提供足够的哈希算力，它们也能从自己的存储能力上获利。另外，节点还能提供数据处理这样的服务，从而在已存储的数据上获取额外的收入。无论是否发生在 Filecoin 的区块链上，这些交易都为 Filecoin 的节点带来了收益，进而促进了有用数据片段

持续的存储和分发。

2.8 共识机制

Filecoin 的挖矿做了有用功：之前存储的文件已被证明保留在了网络里，新的文件被添加了进来，新的交易也已被发布。在比特币里，挖矿也提供了发布交易的有用服务，尽管其工作量证明本质上并没有用。这些挖矿其实都是为了确保全网在账本上达成一致。

在本文所描述的 Filecoin 版本中，有用的存储服务层在比特币共识层之上：在解决了工作量证明之后，矿工还要提供相应的可获取性证明。存储服务是有用的，但在共识机制的工作量证明部分上，全网还是浪费了大量的计算资源。可以直接在共识机制中使用可获取性证明来替代这种方式。在这种情况下，为了生成一个区块，首先会要求矿工证明某些挑战片段的可获取性，这些片段是由前一个区块的哈希值决定的，接着执行一个容易的工作量证明。使用这种机制将会改变比特币风格所要求的假设：明确点说，原本的不能让一个敌对节点控制全网 51%哈希算力的要求，被替换成了一个在存储能力和算力之间可调节权衡的要求，而调节方式是两个无依赖的难度参数。不过这种方式有着一个巨大的漏洞。由于二级市场存在的可能性，一个有强大算力的敌对节点就有可能利用二级市场完美的避开可获取性证明的要求，从而只需要和全网的小部分算力做竞争。

与之相反，我们观察到 Filecoin 的存储服务其实是可以构建在任何强壮的分布式账本之上的，比如说基于股权证明的系统[2, 5]，或者任何基于拜占庭共识机制的系统[3, 6]。由于 Filecoin 的目标是让数据更广泛、更便宜的可用，并把浪费的计算资源重新用于有用的任务中，所以在未来的 Filecoin 中，我们提议完全替换掉共识机制中的工作量证明。

3 结尾

我们展示了一个新的加密货币和文件存储网络 Filecoin。Filecoin 可以将数据存储外包给服务提供商的流动分布式网络。Filecoin 激励提供方分配自己的存储资源给所有被请求的数据片段，因为每个这样的片段在将来都有可能成为带有收益的挖矿挑战主体。各方可以随意选择加入或离开网络，这不会损害网络的健壮性。可调节的参数平衡了弹性和波动，复制因素和共识策略。

参考

[1] G. Ateniese, S. Kamara, and J. Katz. Proofs of storage from homomorphic identifification protocols. In *Advances in Cryptology–ASIACRYPT 2009*, pages 319–333. Springer, 2009

[2] V. Buterin. Ethereum <https://ethereum.org/>, Apr. 2014.

[3] M. Castro, B. Liskov, et al. Practical Byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[4] S. King. Primecoin <http://primecoin.io/bin/primecoin-paper.pdf>, Apr. 2014.

[5] S. King and S. Nadal. Peercoin <http://peercoin.net/assets/paper/peercoin-paper.pdf>, Apr. 2014.

[6] L. Lamport. Byzantizing paxos by refifinement. In D. Peleg, editor, *Distributed Computing*, volume 6950 of *Lecture Notes in Computer Science*, pages 211–224. Springer Berlin Heidelberg, 2011.

[7] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[8] H. Shacham and B. Waters. Compact proofs of retrievability. In *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '08, pages 90–107, Berlin, Heidelberg, 2008. Springer-Verlag.