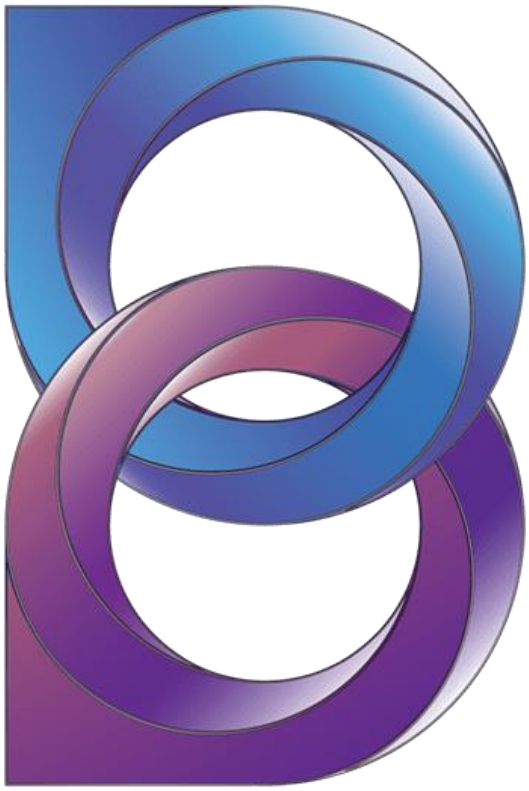
Bismuth
加密货币白皮书



由 Bismuth 核心开发小组编纂

版本号：1.2

发布时间：2019 年 8 月 13 日

目录

摘要

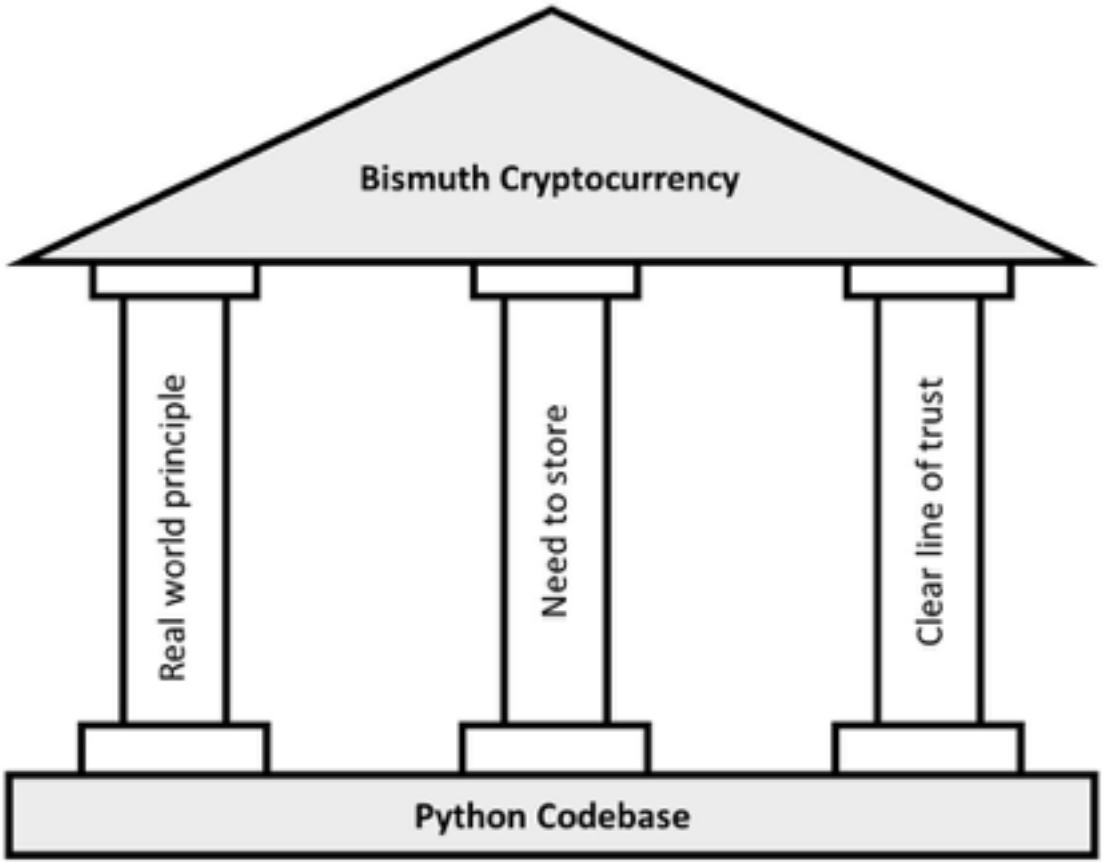
本文是 Bismuth 加密货币白皮书。本文将以 Bismuth 的理念为介绍开始，接着介绍 Bismuth 的三大支柱体系，该支柱体系由如下核心部分组成：插件、用例、代币的提供及奖励模型、加密算法、挖矿算法、尾部祛除块验证、运算和数据项、隐私合约、超级节点和侧链、教育和研究。

介绍

Bismuth 加密货币的初衷是构建一个尽可能简单的、一个全新的用 python 构建的区块链系统。最初，Bismuth 是由一个独立开发者为了学习技术而发起的项目，不久就发展成为了一个功能丰富的加密货币平台，汇聚了许多开发人员、技术支持者、矿池运营商、交易所、多社交媒体影响者参与其中。

Bismuth 支柱

Bismuth 由以下区别于其他加密货币的三大支柱特征构成。



1. 现实原则

有些区块链被塑造得非常完美、非常理想，它们号称自己的代码解决了所有的问题，用户都是专家级别的理解了其复杂的算法逻辑，代码都是没有漏洞的，程序都是由源代码编译完成，用此程序完全没有错误的。它们这种设计方案是注定要失败的，因为它们已经脱离了现实、脱离了用户、脱离了对其的用例。正因为现实（用户、存储、网络）是不完美的，Bismuth 有责任让其实现。

完全没有必要设计一个非常复杂的算法来确保达成如下不可变的共识：它所保护的是不会免费的。复杂的代码应该涉及到真实世界的用例。举个例子，你开车不需要用 12 个超音速助推器，你汽车的引擎已经足够能保证你日常所用了。正因如此，Bismuth 不需要事务 hash，用如下代码就能够验证区块中所包含的事务：

```
block_hash = hashlib.sha224((str(transaction_list_converted) +
    db_block_hash_prev).encode("utf-8")).hexdigest()
```

这就是 Bismuth 的精神：简单、易于理解，有效及考虑周全。

如果我们的目的只是让其在加密货币中得以广泛使用，那么我们完全有必要祛除其中为开发者和专家提供的东西。我们的目的是要让其简单易懂而不是过度构建。以上是 Bismuth 保持尊重的地方，但以下是 Bismuth 持之以很的方向：不是创建一个非常复杂并抽象的系统，然后再去寻找使用场景；而是从现实世界的场景中构建一个最简单有效的，带着充分保障的系统。

2. 可存储

因为在现实场景中，有些数据有必要存储在链上。用户不必需要有像使用 BTC 或 ETH 一样的技巧。有些事物应该被存储在链上的，让我们不要再隐藏这些数据，让我们减少用户的工作。

Bismuth 默认由两项抽象部分构成：运算和数据，用户可以利用这两项来构建任何协议，无论是多么的复杂、完美，仍然可以使其展现出优秀的性能。

但是，存储任何东西在链上的问题，就如锤子所面临的问题：如果你唯一拥有的只是一把锤子，那么你看到所有的问题正如一颗钉子。这就是为什么一些加密货币是如此的关注于以下可扩展的问题：它们想把任何东西都存储在链上，并且是如此的多。Bismuth 的观点的是只存储需要存储在链上的数据，仅此而已。当然，遵循现实原则，在链上存储并不是那么的痛苦。

可扩展性是程序设计之重。为了确保让你存之所需，你需要从头就要开始避免许多扩展性以及长远的问题。当你可以用存储文档的散列（hash）值时来验证其真伪时，为什么你还要去在链上存储所有的文件呢？存储可证明资料而非数据。

用标记点、标识、唯一标记等。

Bismuth 也旨在提供一个可在链中存储的实用框架，当然，其得益于其链的安全性。（参照 Hypernodes-like side chain, Hyperlane）。

3. 明确的信任

加密货币经常宣称自己“不可信”。只是他们忽略了真实世界的规则。

在链上存储的并不完全“正确”。这些加密货币只是在数据一旦存储后不可修改而已，反而需要隐藏一些可信信息：代码中，引导数据中、标记算法中、大量交易对、及服务提供者中。在链上沟通并不神奇，这样并不完美。它们可能有漏洞、虚拟机可能回改变，它们依赖于语言机，而你仍然抱以信任，但你根本不知道你信任的是什么或是谁。

Bismuth 并不会隐藏你所信任事物的事实，并且我们尽量让其更清楚。当你使用 bis 像 Zircodice 或 Dragginator 用于隐私交易时，你信任的是交易者。你仍然可以获得交易历史并且证明交易者所期待想做的。

Bismuth 的执行模型和抽象接口也是如此。你不仅知道你信任的是什么，也可以选择你想信任的。关于协议，你可以尽情的选择你所信任的、所适合你的等去实现你想实现的。在链上并不是什么神奇的不可以改变的，包括漏洞、代理、后门等问题并不是不可修复的。

换句话说，“明确的信任”就是你很清楚你信任的是什么或是谁。Bismuth 不是不可靠的，而是你值得去信任的层面和人的项目。

Bismuth 的价值提议

以下是对 Bismuth 价值的汇总：

- 轻量级：Bismuth 的节点是轻量级的，并不需要强大的 CPU 和大量 RAM
- 对开发者、学者、学生都是友好的。
- 很容易掌握其基础代码并再次开发。
- 支持对用例原型的快速实现
- 容易调整及实验
- 支持多语言的多应用层及客户端的 API

对开发者而言

对于其开发特性，Bismuth 一如既往地遵守以下规则：

- 简单的：它不会拷贝其他链的所有复杂优点，而是精简到其核心使之简明扼要

- 创新的：Bismuth 协议隶属于一个这样的团队：能快速为其测试及添加具有良好弹性、新特性的团队。
- 具有极好的可拓展性和可调节性

Bismuth 是为开发者量身定制的：

- 使用 python 意味着其具备大量的开发者
- 就目前情况来看，python 是大学、学生、学院、大数据及机器学习科学家的开发语言的选择。
- python 不需要预编译，修改即运行。
- Bismuth 可以和多层架构进行通信，从数据库操作、SQL 操作、到不同语言的客户端 API 交互。
- Bismuth 节点由钩子（hooks）及过滤器(filters)构成，可以轻松地用 python 语言编写插件，完全不需要学习新语言。
- Bismuth 钱包由插件构成,所以一系列 dApp 可以无缝的接入到钱包中。
- Bismuth 的抽象传输模型及协议允许任何虚拟程序基于 Bismuth 之上而运行。

一个来自核心开发团队的反馈例子：因为 python 的简易，你只需要几个小时就可以开发几个程序。赋予区块链的智能化的最大难度在于需要研究满足其智能化的所有知识。在很多时候，当我们使用以太坊的 solidity 语言来处理一问题时，我们需要花费大量时间，因为它太复杂了。

拷贝 Bis 的代码仓库是学习 Bismuth 基础及开始使用它的最好方式。Bismuth 的核心概念请参见：GitHub 连接。

Bismuth 的执行模型

当前的 Bismuth 模型与以太坊的完全不同。你可能完全不明白智能合约及 solidity 是干什么的。Bismuth 目前不需要公有智能合约，并且也不需所有节点都运行着同样代码的虚拟机。

尽管这可以被看作是个局限，但其实是个优势。Bismuth 中的一些优势可以弥补以太坊没实现的地方，例如架构中。

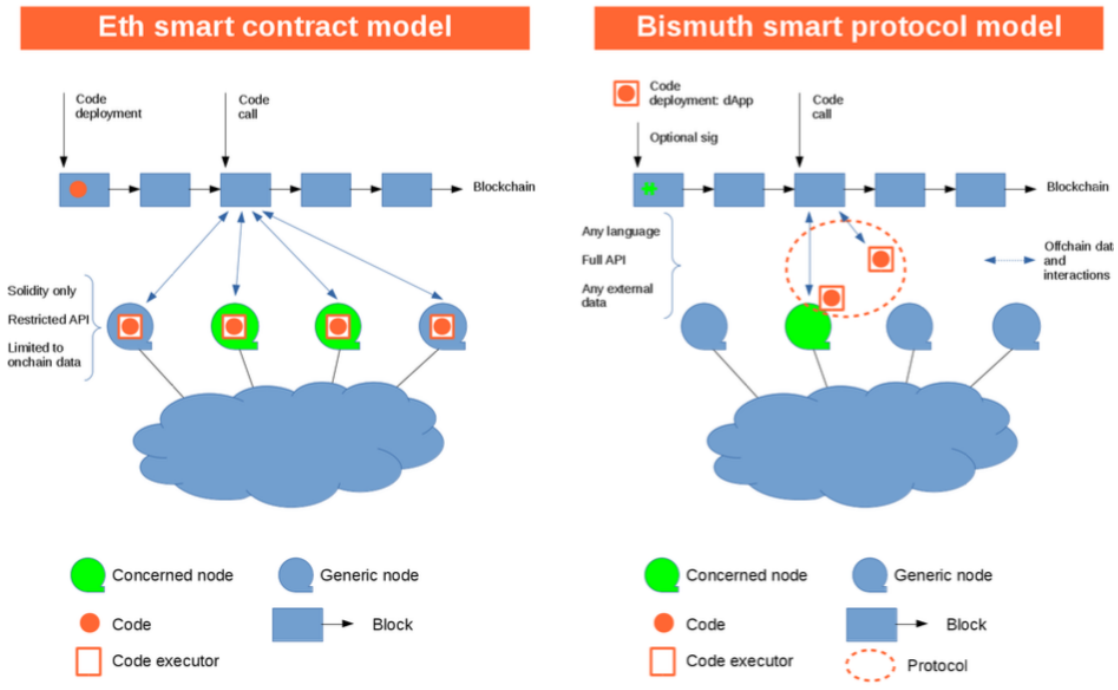
“智能”合约与“智能”协议简单地对比

以太坊的智能合约是用专用语言编写的，存储在链上及在全节点上运行。Bismuth 的智能协议是基于 Bismuth 链之上，且仅运行在与其关联的 dApp 上。

以太坊

- 你需要学习新语言：Solidity
- 有一些缺陷（溢出、可预见度、访问权限）
- 有缺陷的代码可能会造成让用户获取大量的币
- 以太坊智能合约已经发生过一些黑客及可怕的事件
- 智能合约能“操控”资金
- 智能合约永久性的存储于链中，尽管是更新操作也无法阻止其运行，除发开发者提供了一个自杀性的开关。
- 如果有一个自杀性的开关，那么掌控者将会得到合约中的所有资金。
- 所有的合约都是运行于以太坊虚拟机上的某个独立节点，并且消耗着 gas
- 合约无法直接的访问链外资源

虽然以太坊的一些优点和一些用例是不能简单的从 BIS 中复制而来的，但是 BIS 拥有其他优势。



Bismuth

- 不需学习新语言。你可以用任何开发语言编写只能合约或协议，Python 是其原生开发语言。

- 和你通常用的代码相比，没有更多的陷阱。
- 合约不能超支
- 没有 VM（虚拟机）、没有链上代码、没有公约
- 用户可以运行私有合约
- 通过合约，拥有者可以完全控制程序，包括修复及更新
- 如果程序已经发布在合约上了，也是完全可再审计及修改的。
- 客户端只需要调用合约中所感兴趣的部分。
- 私有合约可以做任何事情，包括调用非链上的外部数据

Bismuth 令牌

正如令牌一样被广泛的使用的事物，不会通过持有通用的虚拟机或者代码来实现。如果非常需要一个用例，那么可以写进 Bismuth 的核心代码。Bismuth 的核心开发团队不像 Bitcoin 或者 ETH 一样臃肿，我们可以开发的非常快速。

以下是令牌的用例

- 原生令牌
- 优化的、资源整合的令牌、令牌的数据索引
- 还可以用额外的特性来重载
- 对所有人而言，代码都是被测试过的、公开的、一样的：潜在的漏洞是可标记的，并且可以公开修复的。

为了适配以太坊的特性，当前的部分 Bismuth 令牌兼容了 ERC20。它们不需要授权：你不能让别人用了你的令牌然后又批准。将会很快添加具有很多特性的令牌类型。

Bismuth 令牌使用运算和数据项来创造和制作。其中主要的运算是 令牌：错误，令牌：传输。获得更多 Bismuth 令牌的详情请参见链接。

Bismuth 的“智能”协议

在链上的代码并不是不可变的，你可以完全控制运行在链上任意一个独立的节点上的资金，销毁、锁定或者增加，这正是 bismuth 爱好者称之为“智能”协议的原因。用了引号，那是因为在区块链世界中，没有合约或者协议是真正的智能。称之为智能或者愚蠢，要看是谁开发的。

一个基于 Bismuth 传输的协议可以被称之为抽象数据。所谓抽象数据就得在问题发生时，看两个或多个团体之间同意认定其是什么数据或要做什么。

- 只有在协议中所涉及到的相关客户端才需要知道去读其数据和运行其节点。并不是所有的节点。
- 代码不上链。可以被更新、修复、不会使链崩溃、不会消耗节点资源。
- 在同意方之间就是个“合约”，这是一种理想状态下的公约。
- 每个人都可以通过链上的数据来验证其逻辑，并且可以验证他们所需的其他人的逻辑。
- 协议可以更新、重载、或者将其作为更多先进协议的基础协议
- 协议中可以使用协议……举个例子：协议自身可以定义有效的实现（没有链的哈希）。

查看一些已经存在的 Bismuth 协议，请参见 Github 地址。

Bismuth特点

这一章描述了 bismuth 加密货币的一些特点。

Python 和插件

打算基于 Bismuth 节点之上而开发 dapp 的开发者，被鼓励去使用 Bismuth 的特性被叫做“插件”。插件存在于~/Bismuth/plugins 目录下。Bismuth 插件系统也非常的轻量级，为了便于添加功能，允许对关键事件使用 action 及过滤钩子(filter hooks)。举个例子，想用一个插件来实现锁定钩子操作，仅此需要如下的方法：

```
plugins/900_test/__init__.py:
def action_block(block):
    print(block)
```

想得到更多关于 bismuth 插件的信息，请参见链接。 要激活一个新的插件，需要重启 Bismuth 节点（node.py）执行文件。

用例

1. 实验室用例

这是一件非常容易的事情，当学生们用虚拟机来设置他们自己的私有网络并且将默认端口 5658 改变为其它。通过将本地网络地址（比如 10.0.x.x 或者 192.168.x.x）加入到 Bismuth 的白名单配置中（config.txt）,就可以将实验网络与外部网络隔离。用此网络，学生、研究员、开发者等都可以测试 Bismuth 的新特性或者 dapp，并且你不需要在主网或者测试网络下进行硬分叉。

2. 子链用例

子链用例可以解决可伸缩性及复杂性等问题。作为一个链来说，应该具备应具有的特有属性（出块时间、或有是否有准入门槛）但不应该受限于现有链，比如工作量证明（pow）链。像权益证明（pos）链一样的超级节点是可以运行于工作量证明（pow）链之上的。它们使用了同样的技术，开发者可以为他们的应用使用一个独一无二的链，并且这个链的设置是唯一的、可以定义唯一的传输类型（成为货币或非货币的数据，或拥有两者属性）

3. 事件索引

事件索引是一个存储导致该结果的所有事件的对象或数据模型，而不是存储自身状态。一些简单的 dapp 可以介绍事件索引的概念：请看链接。事件索引可以非常友好的与私有子链结合使用。举个例子。

4. 文件指纹

老钱包使用如下代码即有这样一种特性：可以指纹识别一个或多个文件

```
def fingerprint():
    root.filename = filedialog.askopenfilename(multiple=True,
        initialdir="", title="Select files for fingerprinting")
    dict = {}
    for file in root.filename:
        with open(file, 'rb') as fp:
            data = hashlib.blake2b(fp.read()).hexdigest()
            dict[os.path.split(file)[-1]] = data
    openfield.insert(INSERT, dict)
```

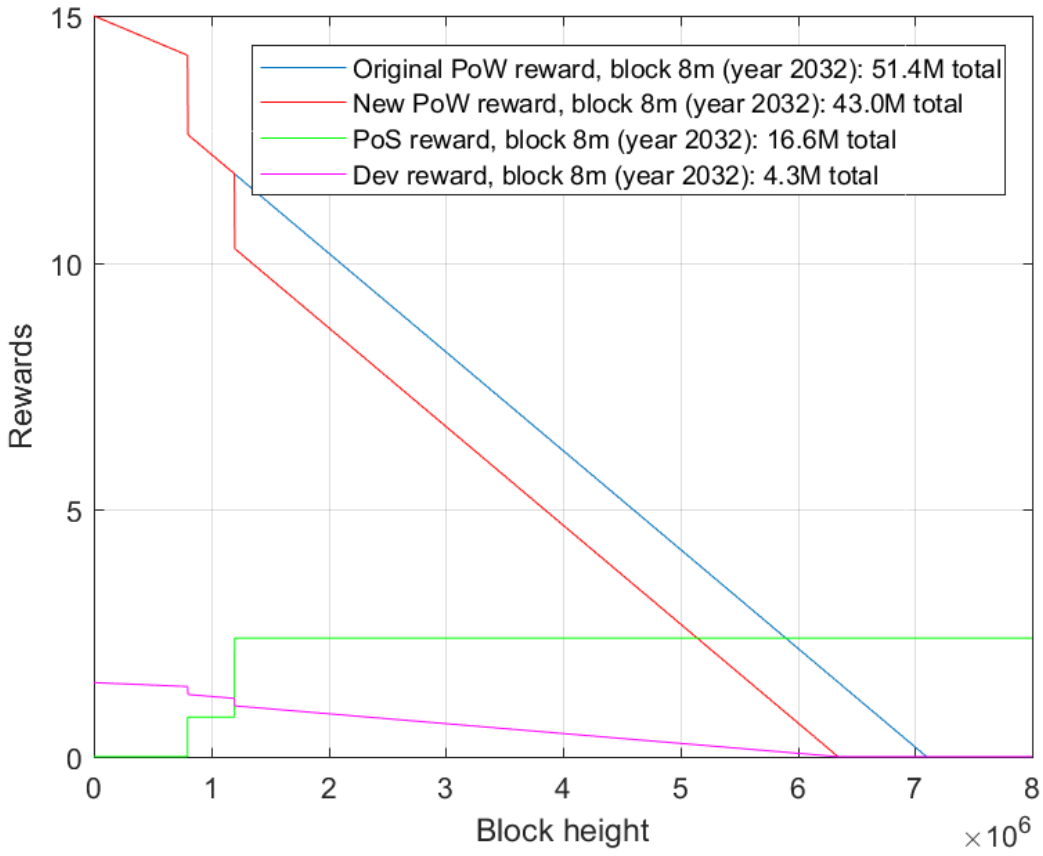
文件的 hash 值被插入到“数据”项中，可以发送给自己或别人。接收者可以用其收到的信息来验证这些文件的真实性。

以下是编撰此白皮书时用了此技术的用例和游戏：

- anon.py，一个网管服务协议私约
- Dragginator，一个基于 Bismuth 区块链开发的收藏类的游戏
- PokaPoka，一个使用 Bismuth 令牌的扑克牌游戏。
- zircodice，一个隐私骰子游戏
- autogame，一个多人的隐私随机游戏

释放币及奖励模型

下面的图显示了 Bismuth 区块链的币释放、奖励模型。在高度为 8,000,000(2032 年，每天大概有 1440 个块)时，BIS 数量为 6390 万个。其中 4300 万个 BIS 作为矿工奖励，1660 万个 BIS 作为超级节点奖励，剩下的 430 万作为开发者奖励（只占矿工奖励的 10%）。如果对此分配有任何改变，将来都会做一次硬分叉，但在编撰此白皮书时，还没有改变此分配的计划。



Bismuth 区块链项目由 2017 年 5 月 1 日开始，下表是此后十年的通胀率表：

Year	Inflation
1	∞
2	93.2%
3	38.9%
4	25.5%
5	18.4%
6	13.9%
7	10.8%
8	8.4%
9	6.6%
10	5.0%

正如上表所示，通胀率在十年内迅速下降到 5%。原因之一是 Bismuth 没有预挖及 ICO：第一个币由创世块诞生（区块高度为 0）。其二，由于该分配模型的原因，通胀率如此高也是自然而然的事情。这种释放币及奖励模型的动机是为了确保 Bismuth 在早期时候能有一个公平的分配模型，同时为了吸引矿工来保证链的哈希率。随着项目的成熟，由于这个快速降低的通胀率模型，可以支持长期持有 Bismuth 代币，同时可以让矿工持续的获得交易费奖励。通过链接，你可以看到 Bismuth 和其他区块链项目在释放代币和奖励模型上的对比：

https://hypernodes.bismuth.live/?p=218

加密算法

当今大多数加密货币采用椭圆曲线算法（ECC），这是一种具有 ECDSA 特性的算法。虽然 ECC 拥有比之前的算法更简短的密钥及更安全，但它是一个新的家族，它也可能存在一些缺陷，显而易见的不安全是基于链式的 ECC。

矛盾的是，Bismuth 采用古老的、众所周知的非对称加密算法：RSA。这是一种诞生于 1977 年的一种依赖大素数算法特性的算法。它已经被广泛使用于网络中的安全 ssh 和 ssl 证书领域几十年了。

钥长度：

- 一个 1024 位长度的 RSA 钥已经可以满足许多中级安全的需求了，比如网站登录。
- 对于需要在多年里都很安全的高安全应用或数据，推荐使用一个 2048 位长度的钥，应该可以保证直到 2030 年都是安全的
- 为了保证数据在未来 20 年都是安全的，RSA 需要大于 2048 位的钥。
- 2031 年后推荐使用 3072 位长度的钥

Bismuth 依赖 4096 位长度的密钥，所以无需担心任何风险问题。

密钥

只有钱包拥有者才知道密钥，当前是以 PEM-base64 格式存储的。

举例如下：

-----BEGIN RSA PRIVATE KEY-----

MIIBgjAcBgoqhkiG9w0BDAEDMA4ECKZesfWLQOiDAgID6ASCAWBu7izm8N4V

2puRO/Mdt+Y8ceywxiC0cE57nrbmvaTSvBwTg9b/xyd8YC6QK7lrhC9N jgp/ ...

-----END RSA PRIVATE KEY-----

公钥

公钥也是用 PEM 格式存储及传输的

举例如下：

-----BEGIN PUBLIC KEY-----

MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAnzgE34oTDlzIPFMsVkNo

foMg9Pm4rG6U8V1fZ/Ewzbtu8UjyvpERblDSaSGBy3C8uZuPpZm/VYTq5KHYJJ6y ... kLYg

WGdQc+MRSkwCwWGQtXECAwEAAQ==

-----END PUBLIC KEY-----

地址

地址匹配公钥的 PEM 的 sha224 散列（hash）值，用 16 进制呈现

举个例子：

3e08b5538a4509d9daa99e01ca5912cda3e98a7f79ca01248c2bde16

签名

Bismuth 用 PKCS1 1.5 版本的签名算法。接收双方的共钥和签名携带着所有的交易信息，并被所有方所验证。

多地址计划

为了拓展 Bismuth 的功能和使用范围，Bismuth 团队已经计划出了几种可选择性的密码学原语。在 2019 年 7 月，这项计划已经规划在发展路线图中了。Bismuth 节点现在已经支持新的 ESDSA 加密算法以及一个新的地址，当然现有的 RSA 算法同样支持这两种地址。

ECDSA 算法通常被用于现存的大多数加密平台，以及用于许多高效率和快速操作中，比如快速签署时间和较小的签名。同时，结合 Bismuth，核心开发者需要尽可能的兼顾 BIP 标准来保证其最佳的兼容性，使得 BIS 能够与基于比特币生态系统的架构无缝连接。接下来将会介绍以“Bis1”为前缀的新地址，这是

一种具有一致性和更友好的 Bismuth 地址。用 ECDSA 算法，Bismuth 兼容传统钱包及助记词钱包。集成与如“Trezor”和“Ledger”的硬件钱包已经变得更简单，最后，使得移动应用钱包支持 Bismuth 成为可能。

对于用户而言

Bismuth 基金将会让用户选择使用 RSA 或者 ECDSA 方案。两种方案都有其各有的优点和缺点，但是这些特点都取决于选择使用哪一种方案的应用开发者和用户。这两种算法方案都会兼容的共存在 Bismuth 协议中，同时可选择性也是面向所有人的。Bismuth 是仅有可以为用户提供选择性的平台，同时提供了高安全性，如果将来在 ECDSA 中发现了后门，因为 Bismuth 在所有交易中使用了 RSA 算法将最不会受影响，并且用户可以立马将所有的操作回滚到 RSA，其中没有复杂的跨链过程。这样的操作在大多数其它使用 ECDSA 的加密平台中是不存在的。

未来证明

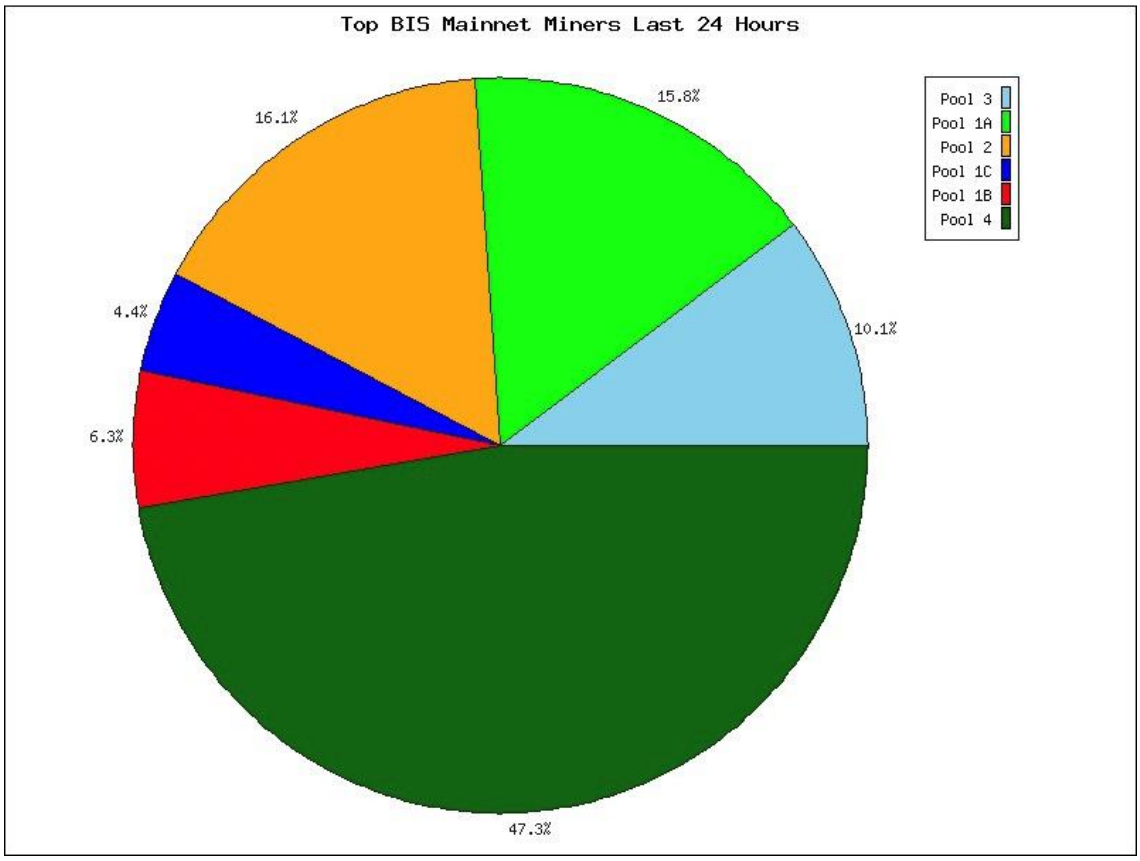
Bismuth 的多方案地址介绍将会为发展中的 Bismuth 带来许多方面的帮助，不仅是拓展其使用领域还是在硬件设备中，抑或是促进其关于地址及格式的功能上都能带来很大的帮助。这是一个具有现代化方案和在建立一个多方案地址方面的突破。在某种情况上来讲，甚至可以突破 RSA 和 ECDSA 两种选择。处理签名和地址的代码是模块化的、可拓展的。举个例子，虽然当前文档没有载明，但是当前节点是支持 ed25519 加密算法的。在不久将来，将会支持越来越多的方案，以此来抵制量子计算对某加密算法的威胁。

挖矿算法

Bismuth 的加密货币项目主网于 2017 年 5 月 1 日上线。挖矿算法基于 sha224, 详情参见：http://dx.doi.org/10.4173/mic.2017.4.1。在挖矿初始，仅支持 CPU 挖矿，但不到半年时间已经支持 GPU 挖矿、同时诞生了第一个 GPU 矿池。Bismuth 在 2017 年 10 月的时候登陆了 Cryptopia 交易所，与此同时新增了许多 Bismuth 账户。在 2018 年 1 月份的时候，交易账户数量是最初的四倍。

因为 Bismuth 拥有简单的算法，所以占用 GPU 较少的显存，同时 Bismuth 网络容易受到 FPGA 或 ASIC 挖矿程序的 51%算力攻击。核心开发组已经很早预见到这种威胁，但决定先解决其他问题，比如提升主网稳定性，而不是开发新功能。超级节点和侧链的引入也是一个例子。

在 2018 年 8 月和 9 月份的时候，诞生了越来越多的 FPGA 矿工，同时算力已经接近总算力的 51%了。下表为当时的算力分布图：



这些大规模 FPGA 矿工是由几个独立的渠道来识别的：Bismuth 网络监控页面，矿池，矿工自己报告的异常，常规交易跳水、与 FPGA 开发人员一起工作的内部核心开发小组。

FPGA 矿工用他自己的账户与在上图列出的 Pool4 之间来回切换。在超级节点成功发布后，核心团队不得不快速行动，并且这种新的挖矿算法已经位列第一了，尽管这种算法没公布在几个月前的路线图中。这种修改后的算法是在 2018 年 9 月基于私密的测试网络开发和测试的。此挖矿算法从概念诞生到发布，仅此用了 3 周的时间。尽管我们的开发速度非常快速，但是交易所及矿池也跟上了脚步，仅用了一周的时间就对节点完成了更新。

是什么让 FPGA 如此高效率的呢？那是因为 Bismuth 挖矿算法仅依赖于处理能力，而不需要内存。在谨慎的研究及测试之后，团队的一名核心开发者 EggdragSyl，想出来如下一些对当前挖矿算法的优化方法：

1.依赖内存

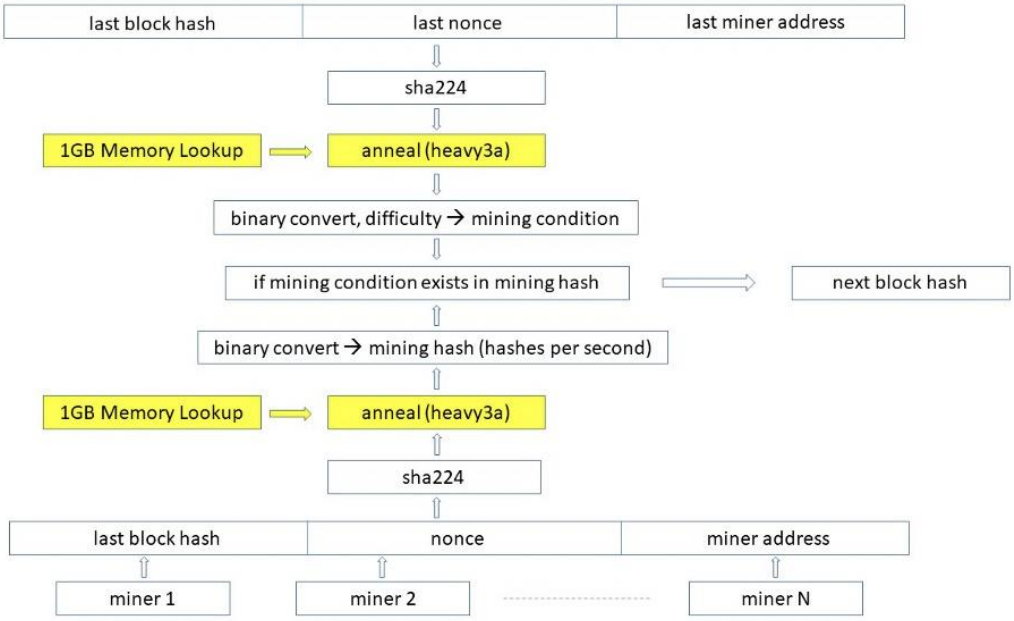
2.加大力度阻止或者惩罚 FPGA 矿工

3.加快节点认证

4.对当前 GPU 矿工做最低的改变来使得矿池快速支持

由此“Bismuth 难度 3”挖矿算法诞生了，并且在分叉后支持挖矿。

在 2018 年 10 月 8 日，高度为 854,660 的块，新的挖矿算法在 Bismuth 主网中发布。虽然以前的 Bismuth 挖矿算法在挖矿上比较昂贵，但是只需要很少的内存。为了让新的挖矿算法抵抗 ASICS 及 FPGA 矿工，需要在内存中引用一个大小为 1GB 的随机二进制文件，如下图中黄色标注部分：



Bismuth 难度 3

由 EggdraSyl 开发的“难度 3”算法简单高效：这种算法所对应的每个测试场景都需要从一个混合查询表中读取一个随机偏差。

这个算法理念可以被其他挖矿算法用来作为抵御相似攻击的附加选择。不管匹配算法是否用哈希现金（hashcash）与否都是无关紧要的，bismuth 没有用。最后测试的哈希状态是 32 位的词向量。因为这是一个哈希结果，可以被当作一个随机向量，可以存储任何东西，甚至你不能反转这个过程(这是哈希的一个特性)，查询表依然存储着随机数据。对于每一个场景，额外的操作是将 XOR 传输到一个哈希输出，被赋予一个从查询表中得到的随机向量，伴随着哈希所定义的下标开始，所以是一个随机的、不可预测的、不可定位的状态。XOR 的哈希状态被认为是复杂匹配函数的输入向量。

- 这种改变并不会影响找到一个好的候选方案的概率。
- 并没有改变哈希算法
- 并没有改变复杂匹配算法
- 对于每一种适用场景，都需要从一个随机下标中读取约 8 个词
- 矿工需要在全时段内保存一份查询表数据到内存中

这是一种可以应用到其他加密货币货币中的通用调整。

中长期规划

核心开发团队依旧钟爱 FPGA 和集成芯片矿机挖 Bismuth。

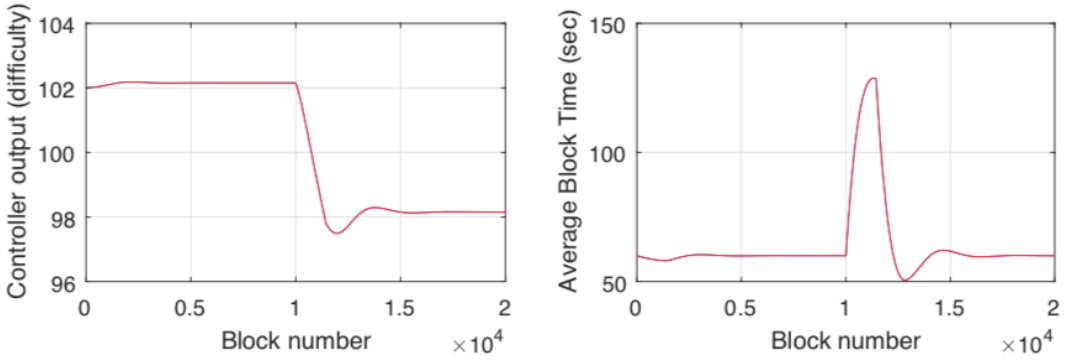
- 这是工作量证明币种可以保护其网络的唯一方式。纯 GPU 挖矿币种总是受到来自 nicehash 或者相似的租赁哈希算力的攻击威胁。
- FPGS 和 ASIC 矿机的算力功耗比优于 GPU 矿机，所以你将会拥有更多有效的算力，将会拥有一个可接管更多资源的更安全的网络。

仅当挖矿设备是大量可用的才成立。这并不是由某个独立的、拥有大量定制或专有的矿机来操作的。

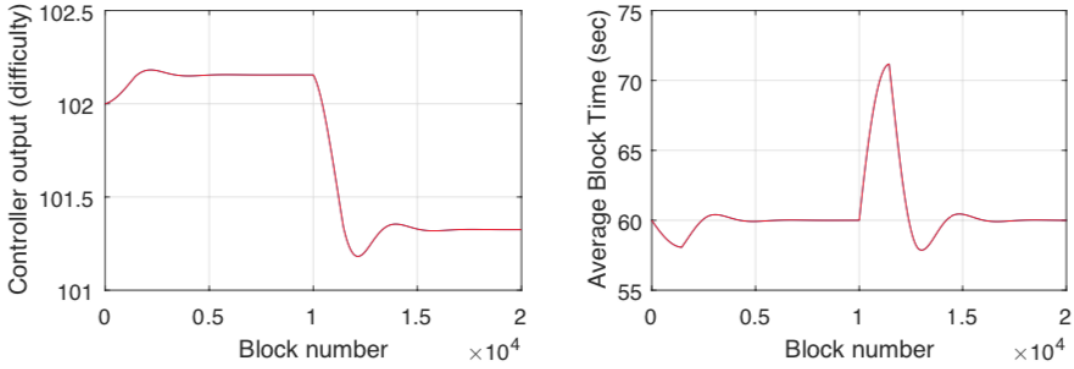
下一步将介绍几种挖矿方式，这样每个人将会有一个公平的机会去挖矿、获得盈利、并且能够为 bismuth 网络的安全而贡献（CPU GPU FPGA ASIC 挖矿）。这样也能让更安全的算法更新再次出现。

链的安全

Bismuth 区块链用了一种叫做反馈控制策略来计算挖矿中的难度，详情在 MIC 中参见：doi:10.4173/mic.2017.4.1 。Bismuth 以这样的反馈控制方式结合最长链规则来达到共识。用这种方法来对比其他方法是如何来决定总算力共识的呢（选择一条拥有最多算力的链）？在 MIC 报告中，图 7 的动态仿真模型回答了这个问题。考虑到以下情形：1）.算力水平稳定在 102，24 小时的平均出块时间稳定在 60 秒以内 2）.在块高度为 10,000 的时候，可能有矿池掌握了超过全网 25%的算力来中断原有链来获得一个更长的链。将会有两条竞争链：1）掌控全网算力的 25%的链 2)在块高度为 10,000 时，主网算力低于 75%的链



上图展示了在算力为 25%时，反馈控制及算力难度是如何反应的。难度从 102 降低到 98，同时出块的平均时间上升到了 130 秒，然后再次回到 60 秒。在这个例子中，生成 20,000 个块所花费的时间是 14.87 天。

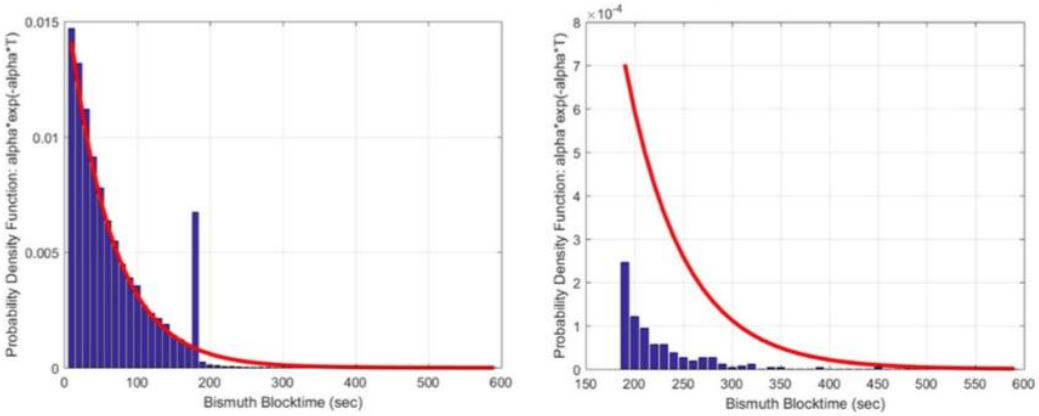


以上图片展示了另一条算力为 75%的链的回馈控制和难度曲线。难度从 102 降低到 101，同时出块平均时间从升到 71 秒，然后再次回到 60 秒。在此例子中，达到块高度为 20000 所用的总时间为 14.02 天。为什么 75%的链要快于 25%的链呢？那是因为在块时间（71 秒对比 130 秒）中的调量小于大算力的链。在这个例子中因为 75%算力的链诞生的 20000 个块比 25%算力的链要快，75%算力的链依旧会成为最长的，同时 Bismuth 中的规则决定了 75%这条链为胜者。从以上仿真模型中可知，当一条拥有最大算力的链将会成为最长的链来以防链的分叉。换句话说，Bismuth 所选择的反馈控制算法同样可以达到其他加密货币中所用的算力为王的算法的效果。难度调节方式及最长链规则是 Bismuth 的独创规则，同样也可以被适用于使用在其他基于总散列工作的共识。

去尾区块证明

概率分布函数（PDF）展示了很多加密货币的出块时间分布情况，比如比特币，他们都有很长的尾部，意味着将会有很小的概率（不为零）能够造成花费很

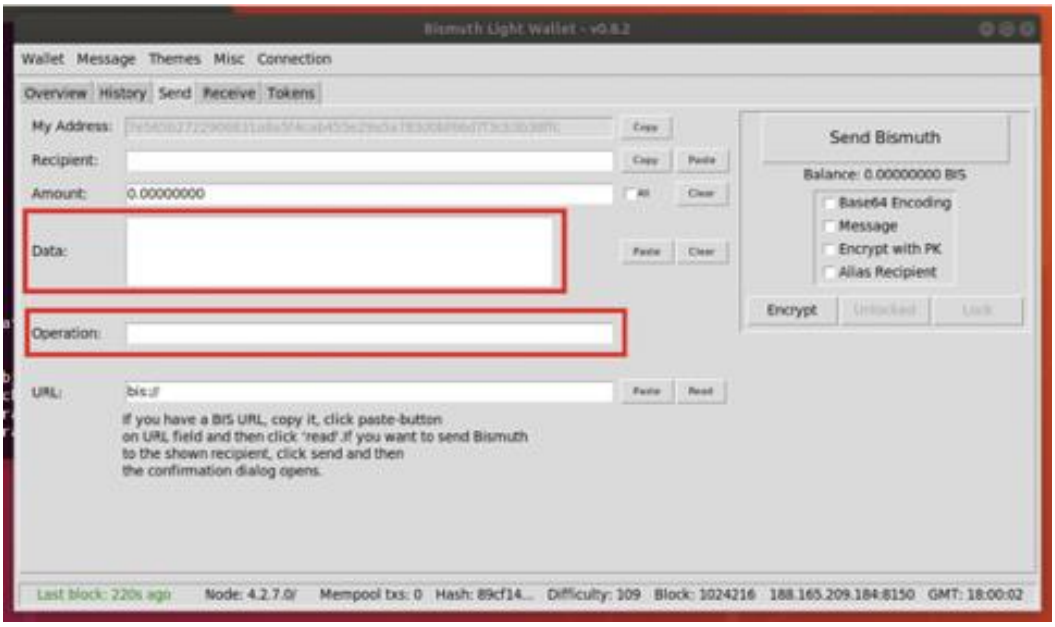
长的时间来诞生一个区块，即使矿工的算力固定或者增长。如此长的出块时间是个问题，有如下两点原因：1).不希望事务处理时间过长。事务处理时间比期望的出块的平均时间还要重要。2).区块链回馈控制算法不能区分长时间的尾区块时间和矿工抛弃的算力。因此，长尾区块时间将会很容易导致算力的不期降低，同时也会造成系统的不稳定性。



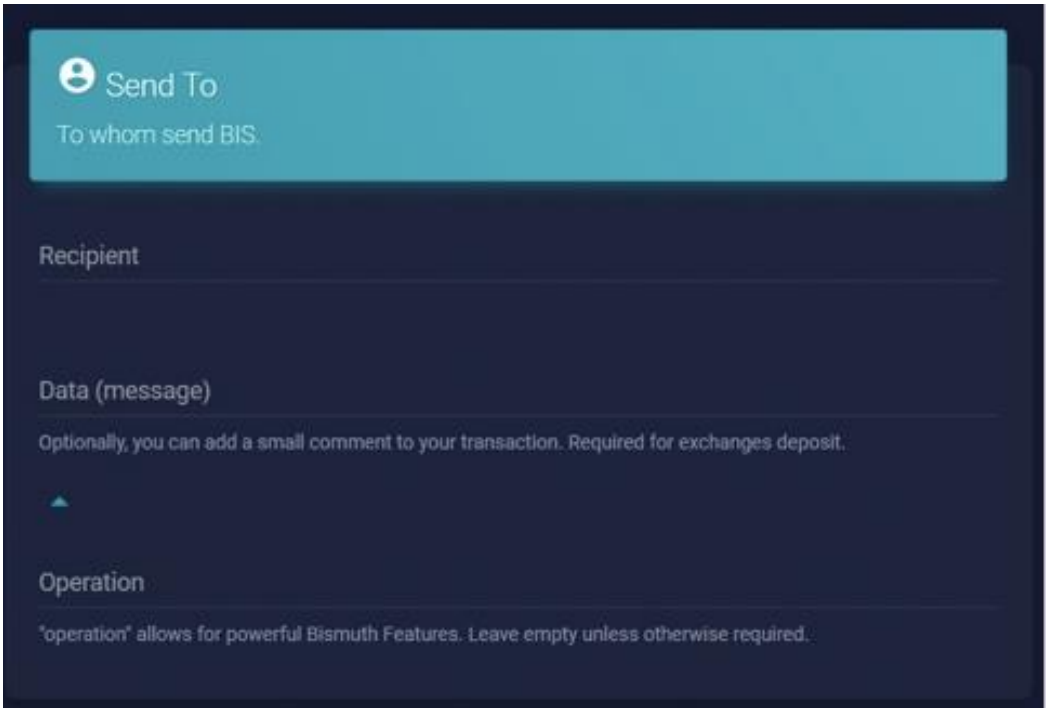
以上图展示了在 Bismuth 中实现了去尾区块后的结果。左图展示了没有使用尾部去除代码（红色曲线图）和使用了尾部去除代码（蓝色柱状图）的情况。右图展示了出块时间大于 180 秒后的情况。正如上图所知，Bismuth 的长尾出块时间明显减少，可以确保即时执行事务。通过以下原文章可以获得更多的实现细节：J. Kučera and G. Hovland, “Tail Removal Block Validation: Implementation and Analysis”,http://dx.doi.org/10.4173/mic.2018.3.1

运算和数据字段项

Bismuth 为 dApp 开发者提供了两项字段项：操作和数据字段项。在下图中用红色方框的标注



Tornado 钱包也支持这两项字段，如下图：



运算和数据字段也可以被用作于编程中，例子如链接中所示。代码如下所示：

```
from bismuthclient.bismuthclient import BismuthClient
if __name__ == "__main__":
    client = BismuthClient(wallet_file='wallet.der')
    if not client.address:
        client.new_wallet()
    client.load_wallet()
    print(f"My address is {client.address}")
    txid = client.send(recipient=client.address, amount=0) # Tries to send 0 to self
    print(f"Txid is {txid}")
```

这段代码展示了向自己发送了 0 个 BIS。对其他账户发送 BIS 只需要将 client.address 字段改成对方的账户地址，如：

recipient=“9ba0f8ca03439a8b4222b256a5f56f4f563f6d83755f525992fa5daf”

将操作和数据整合的方式如下代码示例：

```
txid = client.send(recipient=
    "9ba0f8ca03439a8b4222b256a5f56f4f563f6d83755f525992fa5daf",
        operation='dragg:transfer', data='draggon_adn')
```

这种使用操作和数据的方式与其他加密货币不同。正因为现实原则，Bismuth 项目承认需要在链上存储数据，而不是让使用者知难而退。相反，Bismuth 使用其作为元数据可以让其服务更高级的操作。这就是 Bismuth 的精神所在：易读写，运行的节点不需要知道链上的数据和操作是什么，只需要基于其运行的程序知晓就行。同时操作或数据项可以将节点（基础层的、传输的、真实性的、不可变的、抽象的数据）和分布式程序（使用其数据的、解释的、运行的、更高级的操作）分离开来。

私约

以上所描述的部分,可以用其他两个字段来描述：”数据”和“运算”字段。这两种字段可以用不同组合来创造一些私约：

1) 私 = 加密数据，比如加密消息
2) 私 = 非公开合约代码
3) 私 = 用抽象传输和加密消息来加密的收件方

抽象传输

BIS 的其中一项优势为允许抽象传输。这些具有 0 个 BIS 的交易中的数据，只有运行其相同协议的 dapp 才知道其含义。“运算”项被用作于一种“命令”运算。这种共识是使用如“class:operation”的简单字符串格式来定义的，正如一种命名空间一样。同时这些字段持有较短的数据。开发者可以自定义操作，但是要避免其协议中使用其他已经使用的命名空间，已存在协议如链接所示。

BIS 的传输费用是混合的，同时仅依赖于供填数据项的字节长度。
费用 = 0.01+len(数据项/100000). 不鼓励在链上存储数据，在将来也会被禁止。开发者应该限制存储数据到最小，同时用侧链或者 dapp 到 dapp 的方式来存储所需数据。

超级节点和侧链

当越来越多的主节点发布时，就会证实 Bismuth 的超级结点是具有革命性的。

Bismuth 超级结点作为实验室和验证 Bismuth 的能耐

随着 Bismuth 的成长和团队经验的增长，结点的弱点也会渐显。去测试或改变被很多人或组织正在使用运行中的区块链是不可能的，但是将超级结点作为实验室来测试一些新技术、应用库或者算法是很容易的。在不久后，基于核心代码的 Bismuth 的超级结点将会包含几项新的技术层来供开发者使用。

超级结点还演示了 Bismuth 的抽象事务的集成和使用。超级结点就是 Bismuth 所提供的抽象层和开源的最好应用实例。

网络价值

Bismuth 超级节点的目标是给网络带来附加价值。一些基本的主节点的实现只是标记或者”ping”答案。

Bismuth 超级结点是在一个不同的层来运行的。它们是在其独有的链上运行，不是用货币而是用指标 - 关键性能指标（KPI）。这两种方式是完全不相关的，并且是完全不同的方式来独立运行的。超级节点是运行在权益证明（PoS）的链上，没有预挖，同时链之间没有竞争。超级节点在监控和存储他们的 KPI 到 PoS 链上。正因为超级节点并不是工作量证明（PoW）链的一部分，所以它们不会增加额外的攻击风险。

合格的超级节点是由存储在 PoW 链中的不可变信息而来。PoS 和 PoW 节点的优质指标和不好行为都会被记录下来，并且是在 PoS 链中是不可变的。

这两条链的独立性保证了如下两点：

a/ 操控网络变得更加困难，因为需要用完全不同的方式去改变两条链。

b/ 坏行为和尝试欺骗等手段都会以完全独立和不可变的方式来记录。你不可能在欺骗的时候不被注意。

Bismuth 也许是第一个集成独立的 KPI 侧链来监控网络和确保参与者公平竞争的加密货币。

Bismuth 的 PoW 与 PoS 混合技术

Bismuth 的协议是用 PoW 与 PoS 混合开发的。Bismuth 利用了 PoW 与 PoS 的各自优势，同时避免了其他混合方法的陷阱。PoS 层不是主要共识的组成部分，而是作为 PoW 链的公正观察者，同时因为坏参与者的存在也可以被用作于核心指标。

PoS 监控着 PoW 的参与者，同时 PoW 决定了谁可以成为 PoS 的参与者。这就好像衔尾蛇，每一条链互相制约着对方。从安全来看，这也是一个大的进步，优势在于需要两条链被同时、以同样的方式来攻击。因为两条链的运行机制是不同的，这就是一个史诗级般的任务。

侧链在未来的使用

超级结点的这两种低耦合的链具有很多优势，同时也具有一些未来使用场景。举例如下：

- 没有激励，相反的是在创造更多的块或者阻止他们来伪造。
- 现在来讲，只有一条 PoS 链，超级节点之间有他们自己的规则。但是任何其他使用 PoS 的链都可以被添加到 Bismuth 网络中，同时不用预加载主 PoW 链。你可以把它们当作侧链，这样他们会显得更灵活。
- 因为这些链之间非常低耦合，这样的协议被用作于其他加密货币中。几乎任何用了 PoW 算法的加密货币仅用了像 Bismuth 的超级节点层，就可以从其混合应用层受益，当然也包括其灵活性的侧链和子链。

当然，Bismuth 用其实用的代码来继续开拓新领域。Bismuth 的目标是让其超级节点能够成为轻松构建侧链的框架。每一条链都可以拥有其独有的规则、出块时间、费用（或者 0 费用）、存储、同时由 BIS 主链保证其安全性。

技术小觑

从一个简单的概念证明,Bismuth 代码发展成为一个全功能的节点和客户端代

码库。一些技术的保留是为了保证现有网络的兼容性。超级节点不受其限制，所以超级节点可以从开始使用一些非常现代的应用。

异步输入输出

Async/Await, 协程、是现代 python 一大优势。这样使得可以编写高效、易读的异步代码。不再需要在程序中编写大量的线程，也不需要处理大量的线程锁以及困难的调试程序。也不再需要为了并行程序争夺同一个数据库文件而挣扎。不再有臭名昭著的全局解释器（GIL）锁限制。超级节点的核心用了 Tornado 服务器和客户端，它们之间用异步回调函数通信。这个从其代码来讲即是在性能上的很大提高（几乎不使用任何资源负载）也是其安全性的提升。

Protobuf

Google Protobuf 是一种轻便高效的序列化协议。它几乎支持所有的语言，所以被选作为底层交换格式。这种结构不是详细的文本编码，所以 Bismuth 超级节点将会使用 protobuf.

优点：

- 快、低开销
- 包小
- 许多高效操作
- 跨平台及多语言兼容

加密原语

哈希

超级节点使用 blake2 加密原语。它们具有快、安全、并且具有变量输出长度。

密钥和签名

超级节点地址用了典型的 ECDSA 加密曲线算法。

链耦合

两条链之间是如何交互的呢？PoS（超级节点）是如何使用 PoW（节点）的呢？每一个超级节点运行了一个典型的 Bismuth（PoW）代码。它具有对 PoW 账本和节点状态的访问权限，比如对等对象，共识，区块高度。这些都是只读权限。超级节点仅此是一个观察者。从以上数据来讲，超级节点可以有如下特点：

A/ 得到一个安全、不可变、共享有效的节点列表。在每一轮新的开始，超级

节点都需要一系列节点来选出每一轮的陪审员。这系列节点是由 PoW 链来从以前确定的非常稳定的节点中选出。这是由超级节点拥有者的有效注册来构成的。这个列表不能由 PoS 层操纵。

B/ 通过 PoW 对象来收集指标。在任何时间有新对象加入 PoW 节点，都会留下如下信息：版本号、块高度、ip 地址、链接时间、共识状态、连接状态……同样的操作如回滚一般。即便是一个错误的连接也是值得被记录下来的。这些都是超级节点能够收集和记录在 PoS 链的规则。一旦由超级节点汇成一个环形，这些规则就可以被用作来评估 PoW 参与者的状态，同时也可以将参与者划分为“好”或者“坏”的名单。

PoW（节点）是怎样使用 PoS（超级节点）的呢？

同样的方式，PoW 也可以使用 PoS 所提供的数据：

- 好或者坏的参与者的动态名单
- 关于当前网络高度的可靠信息

这将有利于节点来避免坏的代表参与者-云中的假冒节点或者针对挖矿节点的假节点-也包括在坏区块中的旧节点、被遗弃的、在老版本中的节点。所以，PoS 并不是 PoW 进程的一部分，因为这样只会增加复杂性和提供更多的攻击目标， 而是 PoS 链给予了公正的被用来标的 PoW 和 PoS 参与者的额外数据。

KPI 指标

这里有许多指标需要遵循。超级节点的角色就是精准的收集这些指标，所以团队就可以分析和决定哪些可以用或者怎么用。这些指标可以随着时间而进化。所以可以追踪激活的指标以及与指标相关的东西。Bismuth 开发者认为这将在开始时精心设计，然后进化成治理参数。超级节点和（或者）节点的拥有者愿意投票给各种观察者和层级，以便减轻观察坏行为的负担。在另一个文档中展示了这些精准的准则是如何使用的.

超级节点的费用

由于主节点的机制原理，锻造的节点将会得到由它所锻造的每一个区块的奖励。这就意味着将会激励舞弊行为，因为如果你获得更多的区块或者抵制别人的区块，你将有希望得到更多的奖励。因为有 Bismuth 超级节点的存在，以上舞弊操作将不起效果，因为奖励并不是直接依赖于你所铸造的区块，并且任何人都可以看到你所舞弊而在 PoS 链上永久所留下的记录。当然，因为 Bismuth 的计划，

每一个超级节点所有者都会被定期支付。不涉及到随机支付。每一轮结束就会引导支付奖励给每一个活跃的超级节点。超级节点费用可以有几层意思。第一步，它们将由一个私人协议来处理，同时团队来确保其安全。

步骤如下：

- 有个给定奖励金额（可以是固定的或者可配置的参数）
- “活跃”的超级节点将会在期内分享奖励。活跃的含义是，超级节点在合规的超级节点列表中，意味着在收集和发送规则、与其他节点有交互、如果它具有审计身份，应该会铸造一些区块。
- 奖励应该与抵押数量成正比。如果你的超级节点的抵押是另一个的两倍，那么这个超级节点将有两倍奖励。
- 非公有合约将会计算每一个节点的奖励并分发。
- 算法是公开的。

治理

因为 KPI 和级别可能被用作于触发禁令或者行为来抵制参与者，所以这是一种很自然的治理过程。然而在开始给予一个实验性质的项目是不可能。项目团队将会人工分析进程，有助于对有帮助的 KPI 及其用处有个概览。最后，过滤器将成为自治的，并且大量的参数将由超级节点的用户所投票。

超级区块压缩

Bismuth 因为额外的和速度使用了一个双数据库系统。除了标准的全分布账本数据库以外，超级区块存储了所有的压缩数据，这些数据仅此包含了最新 150000 个块的、之前所有交易余额大于 0 的数据。在有些场景下，超级区块的余额用来对比分布式账本的差异检测。当然，超级区块的数据库在节点启动的时候就装载到了节点的内存之中，也限制了硬盘访问数据库来减少系统的负载，同时增加了数据库的反应速度及可用性。

惩罚系统

在 Bismuth 合约系统里，每一个节点都在追踪所有在线客户端的行径。惩罚是针对所有节点的，它们通过一个单独的区块回滚来造成链的切换，其中一半惩罚将会因为诚实而取消同时将交付给最长的链区块。对于被自动禁止的本地合约来讲，未来的节点是遥远的。这一系列规则决定了攻击者不仅应该拥有超过全网的一半算力，也要攻击者连接到系统中的所有节点，并且在同一时间攻击这些节点和控制大部分的节点。即便这样，每攻击一个节点将会使得攻击下一个节点更

难。

镜像区块

镜像区块技术使得硬编码变得非常简单，比如超级节点的奖励，离线抵押奖励及开发奖励。镜像区块存储是不上链的，而是在传统区块之外，由一个不受同步进程所干扰的负号标志的方式而标记。镜像区块永远都不会在节点之间共享，相反的是它们是由合约创建的，并且可选的依赖于区块链数据。这种方法使得它成为标准数据库的不可或缺的一部分，同时简化了像账户余额评估一样的程序。

Testnet

Bismuth 测试网络是由不同 ip 地址组成的网络节点构成的，非常类似于 Bismuth 主网，但是拥有更新的节点。要创建一个测试节点，以下代码参数必须定义在配置文件 config.txt 中：

```
port=2829
version=testnet
version_allow=testnet
testnet=True
```

在测试网络上的挖矿算法比真实情况的难度低。因此，开发者在测试网络上可以很容易的搭建一个本地矿池和矿工来挖 BIS 代币。推荐 optipoolware 矿池。

Regnet

配置一个 Regnet 节点，需要在配置文件中配置如下参数：

```
version=regnet
version_allow=regnet
```

以下命令可以用用来测试 testnet 和 regnet

```
python3 commands.py statusget
```

Reget 的输出结果如下：

```
Number of arguments: 2 arguments.
Argument List: ['commands.py', 'statusget']
Regtest mode
{"protocolversion": "regnet", "address":
"6a8b4990784617730af465a0dfcbb87284bca8b2189e02798d0a5a5f",
"walletversion": "4.2.9", "testnet": false, "blocks": 1, "timeoffset":
```

0, "connections": 0, "connections_list": {}, "difficulty": 24.0,
"threads": 3, "uptime": 131, "consensus": null, "consensus_percent":
0, "server_timestamp": "1549123577.02", "regnet": true}

对于 regnet，就没有必要设置矿工了。“generate”命令可以批量生成带有当前钱包地址的区块。例如，你可以“generate”十个区块，这样你就可以用 bis 在你的 regnet 上测试。除此之外，“generate”一个区块就可以使得内存池事务被生成。

Regnet 很有用，因为它从块高度等于 1 的地方开始，同时不需要同步网络中的其他区块。dapp 的早期可以在 regnet 中测试许多新特性。

教育和研究领域

Bismuth 对于教育和研究领域来讲是一个理想的平台。以下是一些用 Bismuth 主网或者测试网所做的一些研究案例：

- J. Kučera and G. Hovland, ”Tail Removal Block Validation: Implementation and Analysis”, http://dx.doi.org/10.4173/mic.2018.3.1
- G. Hovland and J. Kučera, Nonlinear Feedback Control and Stability Analysis of a Proof-of-Work Blockchain”, http://dx.doi.org/10.4173/mic.2017.4.1

Regnet 对教育背景来说非常有利，每个学生都可以脱离网络在他们的电脑上运行 regnet。

有许多区块链基础技术都可以用 Bismuth regnet 来教授。研究者或者学者在用 Bismuth 开发新功能，都可以联系链接中列出的 Bismuth 的核心开发者：https://discord.gg/4tB3pYJ. 新的功能和特性被 Bismuth 团队测试后，都有可能被使用于 Bismuth 结点中。

未来的规划

Bismuth 的目标是使得核心结点能够良好的被文档记录，同时尽可能的精简和高效，同时鼓励基于 Bismuth 开发的拓展和应用来使用插件系统和独立库。

Bismuth 开发者致力于“加密标准规范”和“系统生态”发展，用现有工具来实现更简单的交互、协议和平台，因此来介绍彼此的“冲突”。采用了新地址格式（比如 ecdsa），比如 bitcoin 的 json-rpc 服务器、bismuth 服务的 docker

镜像都属于这一类格式。另一个例子是，“Bismuth 客户端”的 python 包已经被证实非常易用，已经由一些团队和第三方组织用它做开发，同时也吸引了很多新开发者。

Bismuth 已经成为一个具有很多特性的加密货币了，同时本白皮书所列举的功能已经被实现了、测试了以及正在使用中。在未来开发的新特性，都会被添加到新的白皮书版本中。Bismuth 的核心开发团队有一个开发路线图，请参照链接：https://github.com/bismuthfoundation/Roadmap

总结

本白皮书陈诉了 Bismuth 的概要理论、支柱和特点。Bismuth 的三大支柱是：1）现实原则，意味着 Bismuth 的核心开发团队对新特性及功能都是采取务实的态度，2）“需要被存储的”意味着你只需要存储你所需要的在链上，不能有其他的。同时，遵循现实原则来讲，在链上存储数据也是不被鼓励的。3）可信任线路，意味着当你决定信任什么事或人的时候，你是这道你所信任的是什么。Bismuth 不是不可信任的，而是可以隐式的信任一些层级和人的。

关于 Bismuth 的一些核心特性已经详细的阐述过了。Bismuth 的详细介绍链接和引用已经提供给了读者，里面阐述了 Bismuth 加密货币的详细信息和技术实现。

免责声明

本白皮书中的信息仅此提供参考，不包含任何投资或理财建议。请你在做任何投资之前做好充分地调研。本白皮书中的信息不涉及构成、应该依赖、建议、提供、引诱从事、或者避免从事、任何购买、售卖、任何涉及到投资的活动，同时对其他加密货币怀着敬意。加密货币投资具有不稳定性和高风险性。不要让投资成本超过你可承受损失的成本。

修订版本

- 2019 年 8 月 13 日：v1.2 版本，支持多地址方案、令牌、格式化，更新了未来的发展规划
- 2019 年 5 月 24 日：v1.1 版本，添加了镜像区块部分、链安全，更新了代币的供应和奖励规则
- 2019 年 4 月 3 日：v1.0 版本发布