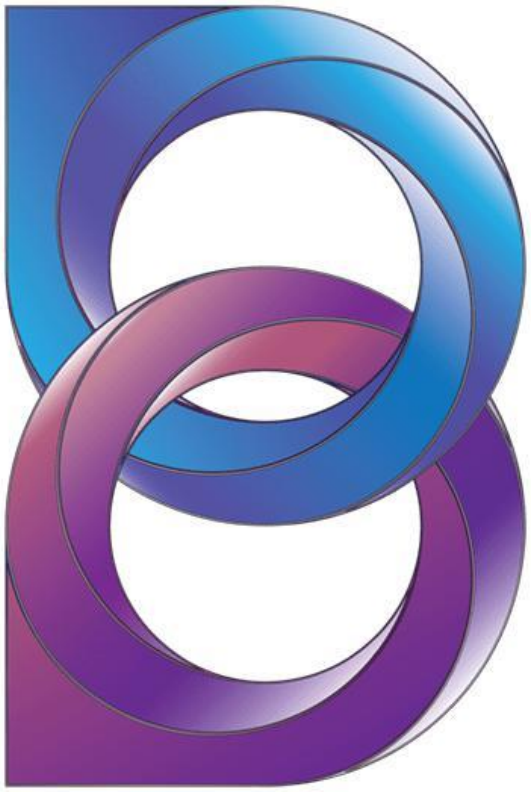
ホワイト ペーパー

ビスマス暗号通貨



ビスマスコア開発者チームによって書かれた
バージョン 1.0
日付: 2019 年 4 月 3 日

内容

抽象

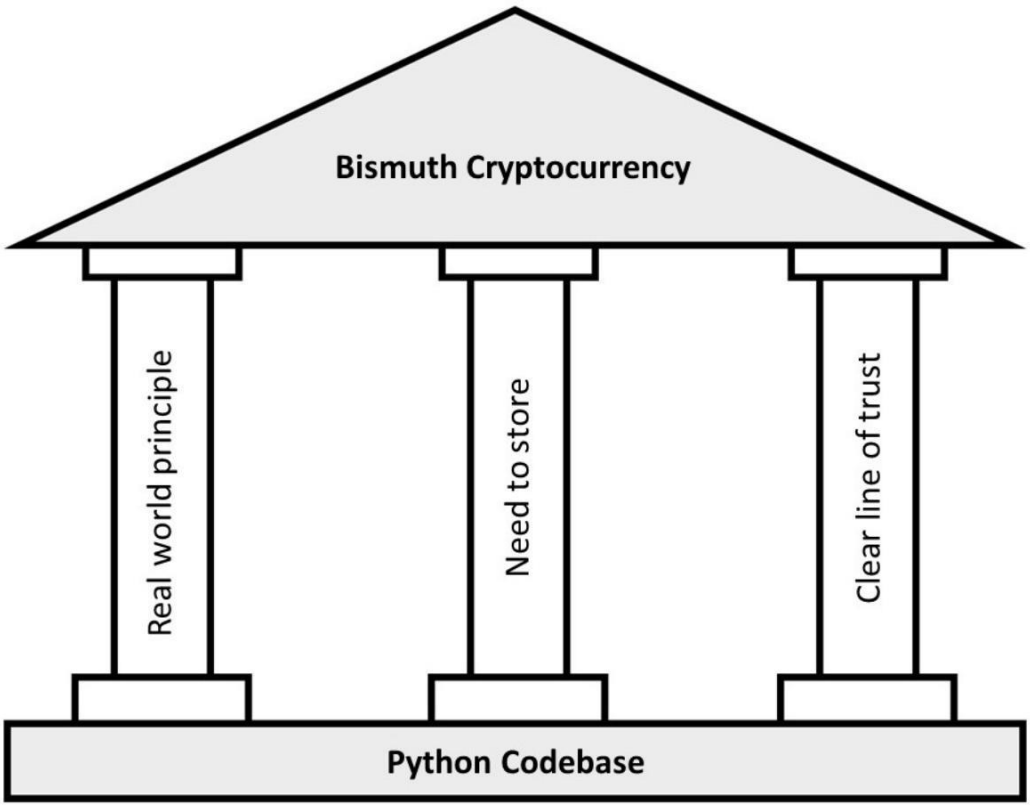
このドキュメントには、ビスマス暗号通貨のホワイトペーパーが含まれています。このホワイトペーパーは、ビスマスの哲学のプレゼンテーションと、プラグイン、ユースケース、コインサプライと報酬モデル、暗号、マイニングなど、いくつかのコア機能のプレゼンテーションに続いて立っている 3 本の柱のスクリプトを記述するところで始まります。アルゴリズム、テール除去ブロック検証、オペラとデータフィールド、プライベートコントラクト、ハイパーノードとサイドチェーン、テストネットとレグネット、教育と研究.

導入

暗号通貨ビスマスの最初の目標は、ブランドの新しい新鮮な Python コードベースから、可能な限りシムプレのチェーンを構築することでした。これは、技術について学ぶために、1 人の開発者によってソナルプロジェクトとして始まりましたが、すぐに複数の開発者、テクニカルサポート担当者、プールオペレーター、交換、ソーシャルメディアインフルエンザのインフルエンザで注目のパックされた暗号通貨とプラットフォームに進化しました関与.

ビスマスの柱

ビスマスは、他の暗号通貨と区別する 3 つの柱に立っています。



1. 現実世界の原則

一部のチェーンは、すべてが完璧にできるアイデアリストの世界に住んでおり、コードはすべてを修正し、ユーザーは複雑なアルゴリズム、バグのない実装、ソースからコンパイルし、ツールを使用してエラーを起こす可能性のあるエキスパートです。その精神で設計されたソリューションは、実際の世界、実際のユーザー、およびツールのユースケースを無視するので、必ず失敗します。現実世界(ユーザー、オラクル、ネットワーク)との顔の間は完璧ではなく、ビスマスはそれを考慮する必要があります。

セキュリティで保護されているものが失敗しない場合に、コンセンサスが不変性に達することを保証する超複雑なアルゴリズムを設計する必要はありません。コードの複雑さは、実際のユース ケースを反映する必要があります。たとえば、車を動かすのに 12 基の超音速原子炉は必要ありません。あなたの通常のエンジンは十分であり、毎日の使用のための十分な保証を与える。たとえば、ビスマスはトランザクション ハッシュを使用しません。ブロック ハッシュは、次のコードを使用して、そのトランザクションに含まれるすべてのトランザクションを検証します。:

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

それはビスマスの精神で、シンプルで、理解しやすく、しかも効果的で、十分に良い側です。

暗号通貨の大量採用を達成することが目標である場合、我々は暗号通貨の開発者や専門家のためにのみのものとして見られているそれらから離れて移動する必要があります。これを行う方法は、それらを理解しやすくし、エンジニアリングを行うのではなく、簡単にすることです。その点ではまだやるべき仕事がありますが、これはビスマスが進む方法です:完璧な抽象システムを設計しようとしてから、一致するユースケースを見つけ、代わりに現実世界のユースケースから始めて、それを達成するための最も簡単な方法を探します保証。

2. 保存する必要がある

現実世界の原則のため、一部のデータをチェーンに格納する必要があります。ユーザーは、BTC や ETH と同じようにトリックを使用する必要はありません。チェーンに保存する必要があるものもあれば、それを隠してユーザーの仕事を楽にしようとするのではなく、保存する必要があるものもあります。ビスマスは、デフォルトで 2 つの抽象フィールド (操作とデータ) をサポートしていますが、ユーザーは、優れたレベルのパフォーマンスを維持しながら、複雑なプロトコルを構築するために利用できます。

しかし、我々はチェーンを持っているので、チェーン上にすべてを格納することは、ハンマーの問題の一種です:あなたが持っているすべてがハンマーである場合、あなたは釘としてすべての問題を見ます。一部の暗号通貨はスケーラビリティの問題に重点を置いている理由です: 彼らはチェーン上にすべてを格納したい、それはあまりにも多くです。ビスマスの哲学は、メインチェーンに保存する必要があるものだけを保存し、それ以上を保存することです。スケーラビリティは、アプリケーション設計の問題です。必要なものだけを確実に保存することで

多くのスケーラビリティと長期的な問題を最初から避けてください。ハッシュのみを保存できる場合に、完全なドキュメントを校正して校正を行う理由は何ですか?データの代わりに校正を保存します。チェックポイント、署名、指紋を使用します。

ビスマスはまた、チェーンの安全性の恩恵を受けながら、チェーンを脇に保管するための実用的なフレームワークを提供することを目指しています(ハイパーノードのようなサイドチェーン、ハイパーレーンを参照)。

3. 明確な信頼ライン

暗号通貨は、多くの場合、自分自身を「信頼なし」と定義します。これは現実世界の原則を無視しています。

チェーンに格納されているものは、必ずしも「真」とは限りません。一度格納されると不変になるはずであり、コード、ブートストラップされたデータ、基になるアルゴリズム、他のピアの大部分、サービス プロバイダーなど、隠れた信頼が必要です。オンチェーン契約は魔法ではなく、完璧ではありません。彼らはバグを持つことができます、仮想マシン(VM)は変更することができ、彼らはオラクルに依存し、あなたはまだ信頼する必要がありますが、あなたは常に何を信頼するか、誰を信頼するか分かりません。

ビスマスは、あなたが時々信頼しなければならないという事実を隠そうとしません。たとえば、Zircodice や Dragginator などのプライベート bis コントラクトを使用する場合は、その演算子を信頼します。トランザクションの履歴を持ち、オペレーターが想定通りの処理を行ったことを確認できます。

ビスマス実行モデルと抽象プロトコルもその方向に向かいます。信頼する必要があるものを知っているだけでなく、信頼できる人を選ぶことができます。プロトコルの場合は、信頼できる実装、自分に合った認証者などを自由に選択できます。これは、可能なバグ、プロキシ、バックドアとそれが間違ったときに修正する方法を持つ、チェーンに保存されているいくつかの魔法の不変の契約ではありません。

ビスマスバリュー提案

ビスマスの価値提案は、次のように要約できます。

1. 軽量:ノードは軽量であり、強力な CPU と多くの RAM を必要としません
2. 開発者、学者や学者のための完璧なフィット感。
3. コードベースは、迅速に処理および開発できます。
4. ユースケースの迅速なプロトタイピングを可能にします。
5. 微調整と実験が簡単
6. 複数の言語で複数のアクセス層とクライアント API を使用できます。

開発者向けビスマス

その開発に伴い、ビスマスは常に次のように試みる。

1. 単純である:他のチェーンのすべての複雑な特徴をコピーするのではなく、取り除き、コアに簡素化するので理解できます。
2. 革新的である:Bismuth プロトコルはコアチームに属し、大きな柔軟性と新機能を迅速にテストして追加することができます。
3. 非常に拡張性と微調整可能である。

ビスマスは開発者向けに調整されています。

1. Python を使用すると、開発者の成長を続けるベースに大きくアクセスできます。
2. Python は現在、大学、学生、アカデミックス、データ、機械学習の科学者に最適な言語です。
3. Python は時間のかかるコンパイル手順を必要としません。微調整とテスト。
4. Bismuth は、直接 DB アクセス (純粋な SQL) から複数の言語の API クライアントまで、インフラストラクチャの多くのレベルでのやり取りを可能にします。
5. Bismuth ノードにはフックとフィルタが付属しており、純粋な Python コードで簡単にプルジンを書くことができます - 学ぶための新しい言語はありません。
6. ビスマスウォレットにはプラグ可能な結晶も付属しているので、dApps は財布にあまり統合されていないように見えます。
7. ビスマス抽象トランザクションモデルとプロトコルにより、事実上すべてのアプリケーションをビスマスの上で実行できます。

開発からコアチームへのフィードバックの例: Python *のシンプルさで、数時間でいくつかの poc アプリを実行できます。ブロックチェーンに才能を持ち込もうとする主な面倒は、必要なすべての研究です。午後のワークショップを行う予定だった時間のほとんどは、あまりにも複雑でした。*

リポジトリを使用したハックは、基礎を学び始め、始める場所です。開発者向けのビスマスのコアコンセプトは、「この GitHub リンク」を参照してください。

ビスマス実行モデル

現在のビスマスモデルはイーザリアムモデルとは大きく異なります。スマートコントラクトと堅牢性で行われる操作を移植することはできません。ビスマスは

現時点ではパブリックな "スマート" コントラクトは必要ありませんし、すべてのノードが同じコードを実行する VM も存在しません。

それは制限と見なされるかもしれませんが、実際にはかなりの強みであり、ETH スマートコントラクトで行われたいくつかのエクスプロイトは、Bismuth のようなアーキテクチャでは成功しませんでした。

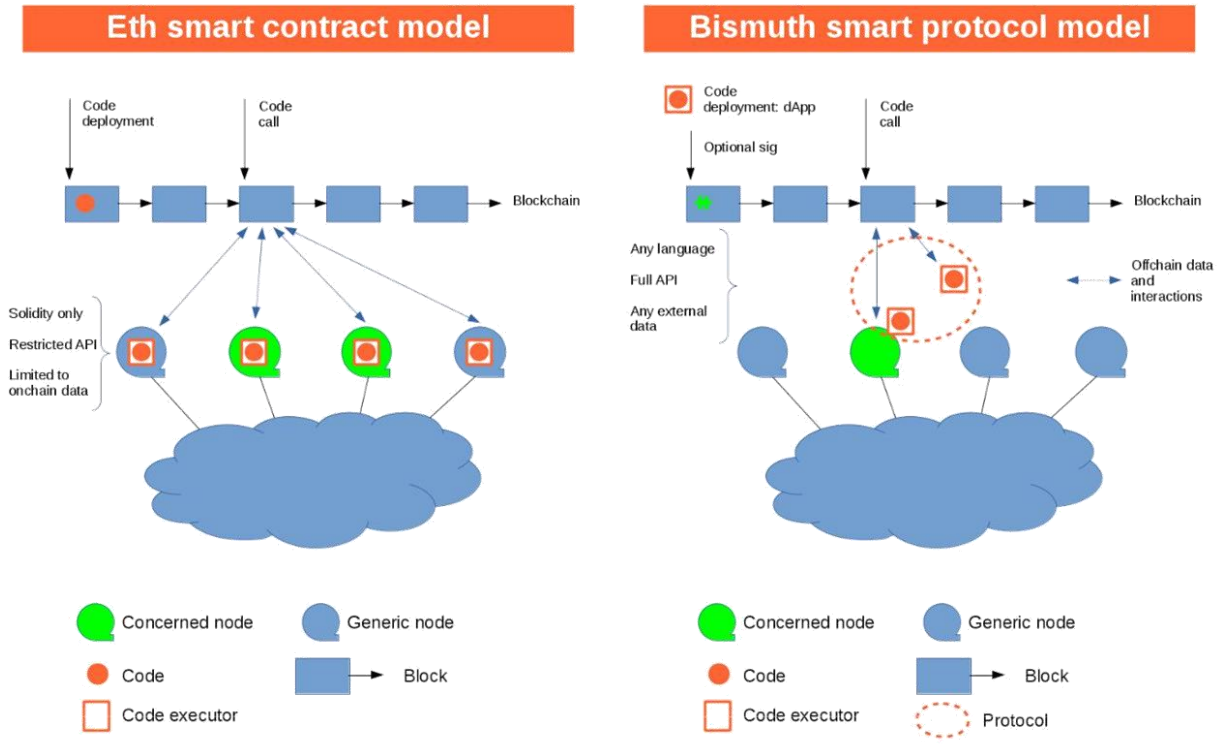
一言で言えば「スマート」コントラクトと「スマート」プロトコル

スマート コントラクトのような ETH は、特定の言語で記述され、チェーンに保存され、すべてのノード**で実行**されます。スマートプロトコルのようなビスマスは、ビスマスチェーンのに実装され、懸念される dApps によってのみ実行されます。

イーテリウム

1. あなたは新しい言語、固体を学ぶ必要があります。
2. いくつかの特定の落とし穴(アンダーフロー、可視性、アクセス権)があります。
3. 欠陥のある契約コードは、ユーザーにコインの無限を与えることができます。
4. イーサリアムスマート契約ですでにいくつかのハックやホラーストーリー。
5. スマートコントラクトは資金を「所有」できる
6. スマートコントラクトは永遠にチェーンに住んでおり、著者がキルスイッチを提供しない限り、停止したり、アップグレードすることはできません。
7. キルスイッチがある場合、所有者は契約のすべての資金を得ることができます。
8. すべてのコントラクト呼び出しは、VM 内のすべての eth ノード (仮想マシン) によって処理され、ガスを消費します。
9. コントラクトは外部リソースに直接アクセスできません。

ETH モデルには、BIS ではレプリケートできない長所とユース ケースがありますが、BIS には他にも用途があります。



ビスマス

1.学ぶべき新しい言語はありません。Python はネイティブ言語であり、ほぼすべての言語でコントラクトやプロトコルを記述できます。

2.通常のコードと同じ落とし穴はありません。

3.契約は過剰に使用することはできません。

4.VM なし、"オン チェーン" コード、パブリック コントラクトなし。

5.ユーザーはプライベート コントラクトを実行できます。

6.所有者は、その後、コントラクト上の完全な制御を持っています - 修正とアップグレードを含む - コントラクト上。

7.契約は、内部作業が公開されている場合、完全に監査可能で検証可能です。

8.コントラクト呼び出しは、その特定のコントラクトに関心を持つクライアントによってのみ実行されます。

9.プライベート コントラクトでは、チェーンオラクルの必要性を排除して外部データにアクセスするなど、何でもできます。

ビスマストークン

トークンなど、広く使用されるものは、汎用 VM とユーザー コードでは処理されません。ユースケースが十分に必要な場合は、ビスマスコアに統合することができます。ビスマス開発チームは、ビットコインや ETH の重量を持っていない、と非常に速く前進することができます。

トークンの場合はそうです。

1. ネイティブ トークン。
2. 最適化された、リソース トークン、インデックス付きトークン db のトークン。
3. まだ余分な機能でオーバーロードすることができます。
4. コードはテストされ、公開され、すべてのユーザーに対して同じです: 潜在的なバグが特定され、グローバルに修正できます。

ETH の用語に一致させるために、現在のビスマストークンは部分的に ERC20 です。彼らは委任を許可していません:あなたは他の誰かがあなたのトークンを費やし、承認することはできません。より多くの機能パックトークンの種類はすぐに追加されるかもしれません。

ビスマス「スマート」プロトコル

チェーンコードに不変の権限を持ち、資金にすべての電力を供給し、それらを破壊またはロックアップさせることができるのではなく、Bismuth は「スマート」プロトコルの概念を支持します。ブロックチェーンの世界ではコントラクトやプロトコルが本当に「スマート」ではないため、引用符が使用されます。それは、それを書いた開発者と同じくらいスマートなコードです。

プロトコルは Bismuth トランザクションに基づいており、抽象データと見なすことができます。これは、そのデータの意味と、イベントが発生したときに何を行うかについて、複数の当事者間の合意です。

1. プロトコルに関係するクライアントのみがデータを読み取り、コードを実行する必要があります。すべてのノードではありません。
2. コードはチェーン上にありません。更新、修正、チェーンを妨害しない、ノードリソースを消費しない。
3. 彼らは合意当事者間の「契約」であり、論理は理想的には公表されている。
4. 誰もがチェーン上のデータ上でロジックを実行し、誰もが必要にとって動作したことを確認できます。
5. プロトコルは、進化したり、過負荷にしたり、より進化したプロトコルの基礎として機能したりできます。

6 プロトコルはプロトコルを使用できます...たとえば、プロトコルは、それ自体の有効な実装 (チェーン ハッシュを使用して) を定義できます。

既存の Bismuth プロトコルの一覧については、「この GitHub リンク」を参照してください。

ビスマスの特徴

このセクションでは、ビスマス暗号通貨のコアフェアの一部のプレゼンテーションと説明が含まれています。

Python とプラグイン

ビスマスノードの上に dApps を構築する予定の開発者は、「プラグイン」と呼ばれるビスマス機能を利用することをお勧めします。プラグインは、ダイレクトリー~/ビスマス/プラグインに存在します。Bismuth プラグインシステムは、非常に軽量ですが、簡単に機能を追加するための重要なイベントにアクションとフィルタフックを可能にします。たとえば、「ブロック」アクションフックを実装したいプラグインは、単純な関数を宣言するだけでいい:

```
plugins/900_test/__init__.py:

def action_block(block):
   print(block)
```

ビスマスプラグインの詳細については、こちらのリンクを参照してください。

新しいプラグインをアクティブにするには、Bismuth ノード (node.py) を再起動する必要があります。

ユースケース

1. ラボのユースケース

学生が仮想マシンを使用して自分のプライベート ネットワークを設定し、既定のメインネット ポート番号 (5658) を別の方法に変更することは非常に簡単です。ローカル ネットワーク アドレス (10.0.x.x または 192.168.x.x など) を使用し、これらを Bismuth 構成ファイル (config.txt) にホワイトリストに登録することで、ラボ ネットワークを外部から分離できます。このようなネットワークを使用すると、学生、研究者、または開発者は、通常のメインまたはテストネットでハードフォークを必要とせずに、新しい機能や dApps をテストできます。

2. 子チェーンのユースケース

子チェーンのユース ケースでは、スケーラビリティと柔軟性に対処できます。特定のプロパティ(ブロック時間、入り口バリア -またはそうでない)を持つチェーンが必要であり、既存のメインネット、作業証明(pow)チェーンによって制限されません。ハイパーノードのような pos チェーンは、パウチェーンの上で実行できます。同じテクノロジを使用し、開発者は独自のチェーンを使用して、アプリ専用の独自のチェーンを持つことができます。

設定を行い、一意のトランザクションタイプ (通貨または非通貨データ、またはその両方) を定義します。

3. イベントソーシング

イベント ソーシングは、現在の状態自体ではなく、オブジェクトの現在の状態につながるイベントを格納するオブジェクト/データ モデルです。ビスマスを使用したサンプル dApps を使用したイベント ソーシングの概念実証 (poc) は、このリンクで入手できます。イベント ソーシングは、プライベート子チェーンとの組み合わせで適切に機能します。たとえば、顧客/プロバイダー/パートナーのネットワークが、共有データベースの運用に合意しているとします。商品、貨物、請求書などの追跡が可能です。子チェーンを操作し、すべてのアクターがノード (ハイパーノードのようなプライベート登録を使用) を実行し、イベント ソーシング poc を使用できます。つまり、アクターは分散データベースとレプリケートされたデータベースを共有し、データに対するすべての変更がイベントであり、不変のタイムスタンプとイベントのソースに加えて、権限を完全に監査できます。

4. ファイルフィンガープリント

レガシ ウォレットには、次のコードを使用して 1 つ以上のファイルをフィンガープリントする機能が含まれています。

```
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)
```

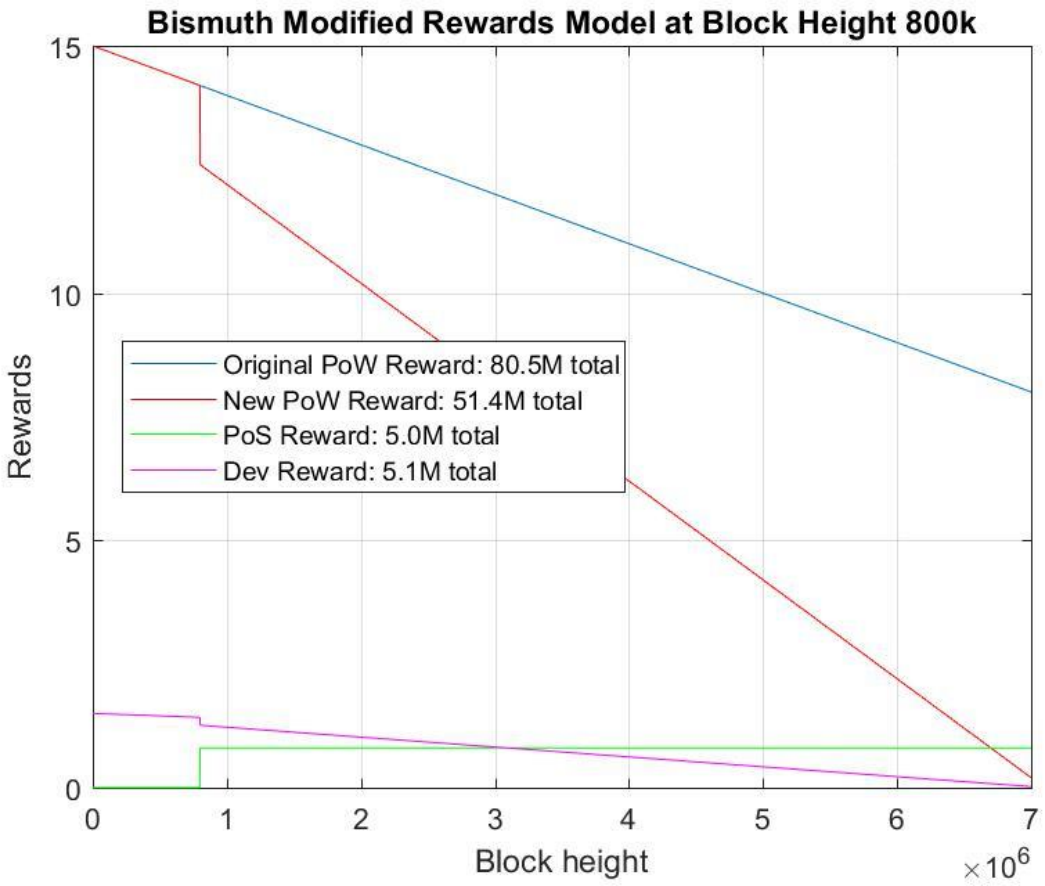
ファイルのハッシュは"データ"フィールドに挿入され、自分自身または他のユーザーに送信できます。受信者は、受信したメッセージを使用して、これらのファイルの信頼性を検証できます。

以下は、執筆時点で実装されたユースケースとゲームの一覧です。

1. anon.py、プライベート契約の匿名化サービスです。
2. ドラジニエーター、ビスマスブロックチェーンに基づいて収集可能なゲーム。
3. ビスマストークンを使用したポーカーゲームサイト、ポカポカ。
4. ジルコディス、プライベート契約としてのサイコロゲーム。
5. オートゲーム、プライベートコントラクトとして実装確率的なマルチプレイヤーゲーム。

コイン供給と報酬モデル

以下のプロットは、ビスマスブロックチェーンのコイン供給と報酬を示しています。ブロックの高さで 7,000,000(2031 年には 1 日あたり 1440 ブロックを想定)、コイン供給の合計は 6,150 万 BIS になります。この金額から 5,140 万人が鉱夫の報酬となり、500 万 BIS がハイパーノードに報酬を与え、510 万 BIS が開発者報酬(10%)鉱夫の報酬の)。このディストリビューションを変更するには、執筆時点では計画されていない将来のハードフォークが必要になります。



Bismuth ブロックチェーンは 2017 年 5 月 1 日に開始され、次の 10 年間のインフレ率を以下の表に示します。

年	インフレ
1	*1*
2	93.2%
3	43.2%
4	27.6%
5	19.6%
6	14.7%
7	11.3%
8	8.9%
9	6.9%
10	5.3%

この表から分かるように、インフレ率は当初はかなり高く、10 年目には 5.3%に急速に低下しています。比較的高い初期インフレの理由の 1 つは、ビスマスがプレミンまたは ICO を持っていなかったという事実です:総コイン供給はジェネシスブロック(ブロック高さ 0)でゼロから始まりました。当然のことながら、インフレ率は、このような分布モデルで最初は大きくなります。このコインサッププリと報酬モデルの背後にあるいくつかの動機は、初期の段階でビスマスの公正な流通モデルを確保すると同時に、彼らのハッシュレートでチェーンを確保するために鉱夫を引き付けることです。プロジェクトが成熟するにつれて、ビスマスを長期的に保有することは、急速にインフレ率が低下し、鉱夫は取引手数料を集めることによって報酬を得るために、短期的に奨励されます。Bismuth のコイン供給と報酬モデルと他のブロックチェーンプロジェクトとの比較については、https://hypernodes.bismuth.live/?p=218 参照してください。

暗号化

今日の暗号通貨のほとんどは、ECDSA 署名アルゴリズムと共に楕円曲線暗号-ECC を使用しています。ECC キーとシグネチャは、以前の暗号鍵やシグナターアルゴリズムよりも少ないビットで短く、潜在的に安全ですが、それらは比較的新しいファミリであり、完全に新しいクラスの欠陥が見つかる可能性が常に高く、すべてを効果的にレンダリングします。ECC ベースのチェーンは安全ではありません。

Bismuth - 逆説的に - 1977 年の出版以来研究され、大きな素数のコア特性に依存している古いが、非常によく知られている非対称メトリック暗号アルゴリズム、RSA を使用して革新します。これは、数十年以来、ウェブ上のセキュリティで保護された ssh と sSL 証明書で広く使用されています。

キーの長さ:

1. 1024 ビットの RSA キー長は、 サイト のログインなどの多くの中程度のセキュリティの純粋なポーズで十分です。

   2 セキュリティの高いアプリケーションや、数年間機密性を維持する必要があるデータの場合は、2048 ビット キーを推奨し、2030 年まで安全にする必要があります。.

3 今後 20 年以上データの機密性を維持するために、RSA では 2048 ビットを超えるキーサイズを推奨しています。.

4 使用後 2031 には 3072 ビットが推奨されます。.

ビスマスは 4096 ビットの RSA キーに依存しているので、リスクを取らないようにします。

秘密キー

秘密キーはウォレット所有者のみが知っています。現在、PEM - base64 - 形式で保存されてす。

例:

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

MIIBgjAcBgoqhkiG9w0BDAEDMA4ECKZesfWLQOiDAgID6ASCAWBu7izm8N4V

2puRO/Mdt+Y8ceywxiC0cE57nrbmvaTSvBwTg9b/xyd8YC6QK7lrhC9Njgp/

...

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

公開キー

公開キーも PEM 形式で保存および送信されます。

例:

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

MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAnzgE34oTDlzlPFMsVkNo

foMg9Pm4rG6U8V1fZ/Ewzbtu8UjyvpERblDSaSGBy3C8uZuPpZm/VYTq5KHYJJ6y

...

kLYgWGdQc+MRSkwCwWGQtXECAwEAAQ==

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

アドレス

キーに一致するアドレスは、公開キー PEM の sha224 ハッシュで、六逆形式で指定します。

例:

3e08b5538a4509d9daa99e01ca5912cda3e98a7f 79ca01248c2bde16

Signatures

ビスマスは PKCS1 v1.5 シグネチャ アルゴリズムを使用します。公開キーと署名の両方がすべてのトランザクションで送信され、受信時に検証されます。

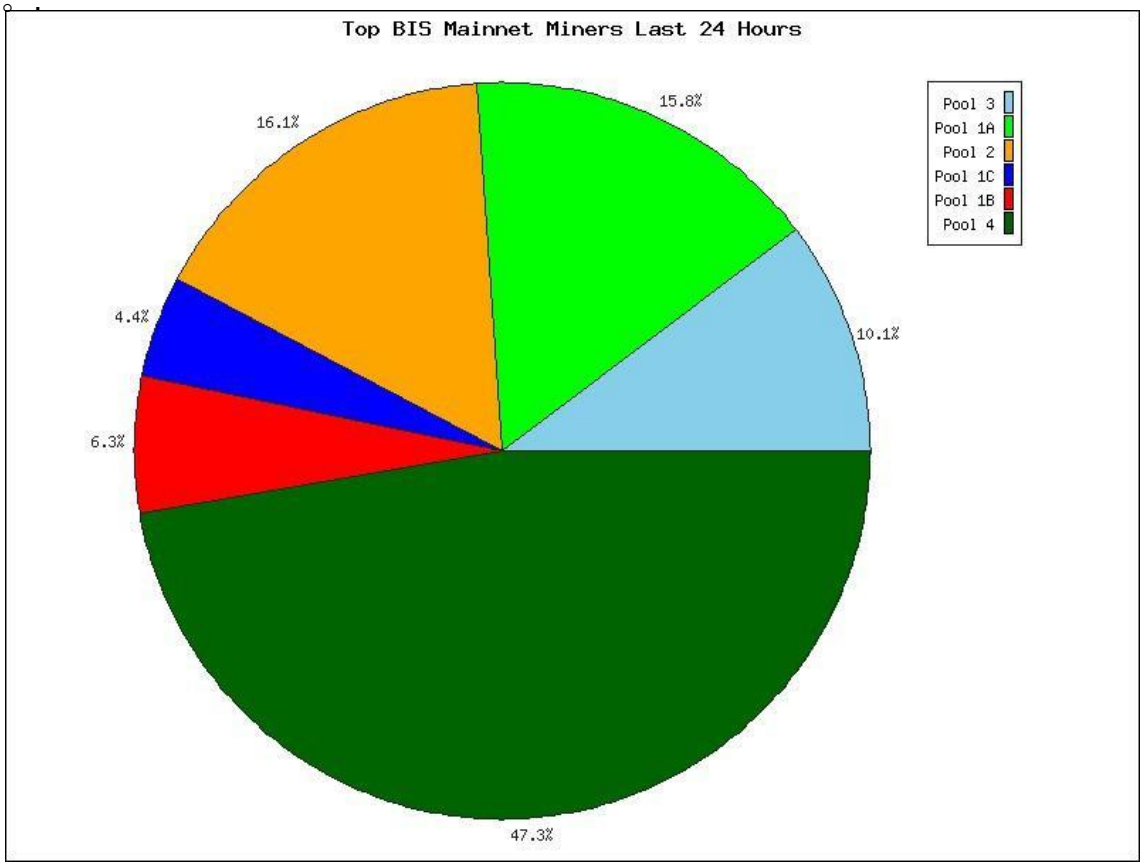
マイニングアルゴリズム

ビスマス暗号通貨プロジェクトのメインネットは、2017 年 5 月 1 日に開始されました。マイニングアルゴリズムは sha224 に基づいており、ここで簡単に説明 http://dx.doi.org/10.4173/mic.2017.4.1。最初は CPU のみでしたが、6 ヶ月も経たないうちに最初の GPU マイナーが登場し、その直後に最初の GPU マイニングプールが登場しました。ビスマスは、2017 年 10 月に Cryptopia の元変更に上場し、ネットワーク上の新しいアカウントが大幅に増加しました。2018 年 1 月までにビスマスのアカウント数は 4 倍に増加しました

為替上場前と比較して。

Bismuth は GPU 上で非常に少ないを必要とする比較的単純なマイニングアルゴリズムを持っていたので、ネットワークは大規模な FPGA または ASIC マイニング操作による 51%の攻撃に対して脆弱でした。コア開発チームはこの脅威を十分に認識していましたが、新しい機能や機能に加えて、一般的なネットワークの安定性の向上など、他の問題に取り組むことにしました。ハイパーノードとサイドチェーンのイントロダクションはその一例です。

2018 年 8 月から 9 月にかけて、FPGA 鉱山が開発され、この採掘作業がネットワーク全体のマイニング電力の 51%に近づいていることが明らかになりました。次の図は、その時点での異なるプールのハッシュレート分布を示しています。:



大規模な FPGA 鉱夫の存在は、いくつかの独立したチャンネルによって識別されました:ビスマスネットワーク監視ページ、プール、鉱夫自身が異常を報告し、定期的な交換ダンプ、FPGA の内部コアチーム研究だけでなく、作業 FPGA 開発者。

FPGA 操作は、自分のアカウントでのマイニングと、上の図のプール 4 の使用との間で交互に行われました。ハイパーノードが正常に起動した後、コア開発チームは迅速に動作する必要があり、マイニング アルゴリズムの進化は、数か月前に公開されたロードマップに配置されていなかったにもかかわらず、優先度リストの一番上に移動しました。変更された

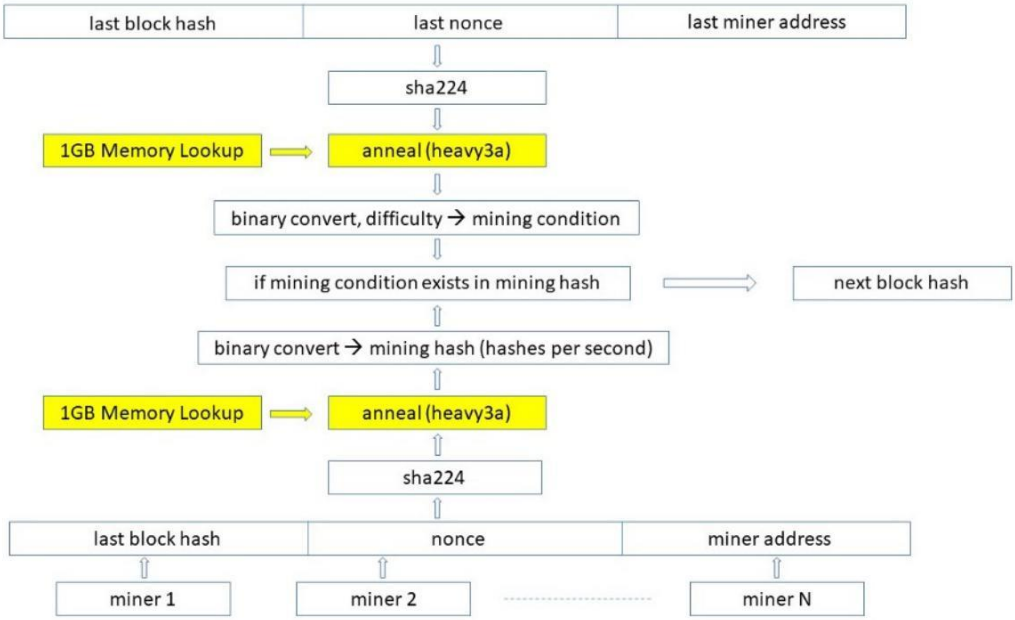
マイニング アルゴリズムは、2018 年 9 月の間に、プライベート テストネットで開発およびテストされました。新しいマイニングアルゴリズムの立ち上げまで、最初の概念的なアイデアから 3 週間もかからでした。この急速な開発ペースでも、交換とプールには、ノードを更新するための 1 週間の通知と時間が与えられました。

FPGA マイニングを非常に効率的にしたのは、従来のビスマスマイニングアルゴリズムは処理能力しか必要としなかったが、メモリが必要ではなかったという事実でした。慎重な調査とテストの後、コア開発者の 1 人である EggdraSyl は、現在のマイニング アルゴリズムに若干の変更を行いました。

1. メモリが難しいです。
2. 特定の FPGA マイナーをブロックまたはペナルティたくさんになります。
3. それでもノードで確認するのは高速です。
4. 現在の GPU マイナーに対する最小限の変更しか必要とされないため、プールによって迅速に調整できます。

「ビスマスヘビー3」マイニングアルゴリズムが生まれ、フォークの後に使用されます。

2018 年 10 月 8 日に - ブロックの高さ 854,660 - 新しいと新しい Heavy3 鉱業アルゴリズムは、ビスマスメインネットに導入されました。以前は、Bismuth マイニング アルゴリズムは計算コストが高かったが、メモリはほとんど必要とされなかった。新しいマイニングアルゴリズムを FPGA や ASIC に対してより耐性にするために、下の表の黄色いボックスに示すように、メモリに 1GB のランダムなバイナリファイルを保持する必要があります。



Bismuth Heavy3

EggdraSyl によって設計された「Heavy3」アルゴリズムの背後にある考え方は、シンプルで効果的です:テストされた nonce ごとに、固定ルックアップテーブル内のランダムオフセットからの読み取りが必要です。

この概念は、同様の攻撃から保護するために追加のレイヤーとして他のマイニングアルゴリズムに適用できます。一致するアルゴリズムがハッシュキャッシュを使用するかどうかにかかわらず、ビスマスは無関係です。テストされる最終的なハッシュ状態は、32 ビットワードのベクトルです。ハッシュ結果であるため、ランダムベクトルと見なすことができます。各 nonce に対して、追加のステップは、ルックアップ テーブルからランダムベクトルを与えられたハッシュ出力に XOR 変換を適用し、インデックスはハッシュ自体によって開始されるため、ランダムで予測不可能な位置で開始されます。結果 – xor ハッシュ状態 - 難易度一致関数への入力ベクトルとして考えられています。.

1. この変換は、良好なカンダテを見つける確率には影響しません。
2. ハッシュ アルゴリズム自体は変更されません。
3. 難易度一致アルゴリズムも変更しません。
4. これは、すべての試みノンスのためのランダムなインデックスから約 8 単語の読み取りを必要とします。
5. マイナーは、ルックアップ テーブル全体のコピーを常にメモリに保持する必要があります。

これは、他の暗号に即座に適用できる一般的な微調整です。*中長期的な考慮事項*
コアチームは依然として FPGA に賛成しており、ビスマス専用の ASIC ハードウェアは、なぜではないのか。

1. これは、PoW コインがそのネットワークを保護できる唯一の方法です。純粋な GPU コインは、常に素敵なハッシュや同様にレンタルハッシュ攻撃のなすがままです。
2. FPGA と ASIC のハッシュ/ワットは GPU よりもはるかに優れているため、より多くのリソースを引き継ぐために必要なより安全なネットワークを使用して、より多くのハッシュを使用できます。

これは、鉱山機器が主に利用可能な場合にのみ当てはまります。これは、単一の操作に何千ものカスタム独自のハードウェアがある場合ではありません。

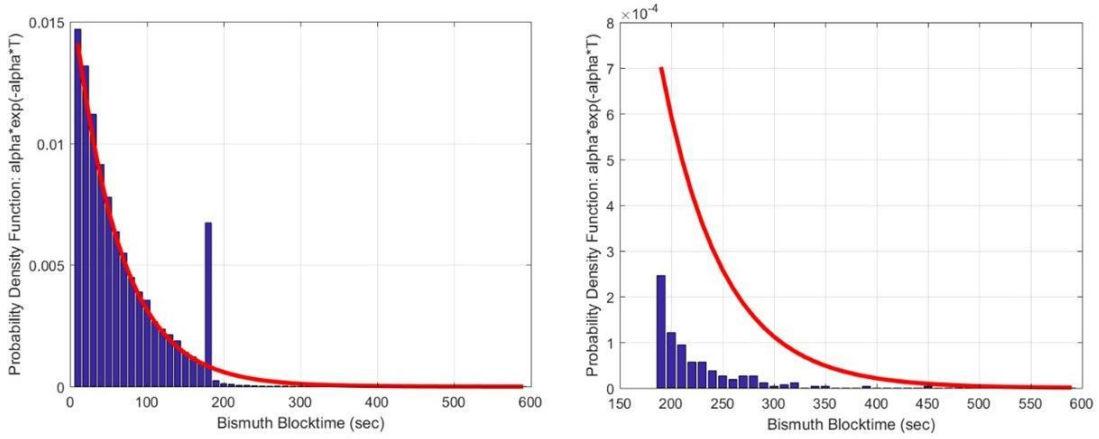
次のステップは、誰もが採掘し、収益性に到達し、ネットワークに貢献する公正な機会を持つことができるように、*いくつかのマイニングチャネル*を導入することです

安全(CPU、GPU、FPGA、ASIC)。また、同様の状況が再び発生した場合に、アルゴリズムの変更を高速化することもできます。

テール除去ブロックの検証

ビットコインなど、多くの暗号通貨におけるブロックタイムの分布を記述する確率密度関数(PDF)は、長い尾を持ち、新しいブロックの生成に非常に長い時間がかかる可能性があることを意味します。鉱夫の計算能力が一定または増加している場合でも。このような長いブロック時間は、次の 2 つの理由で問題です。

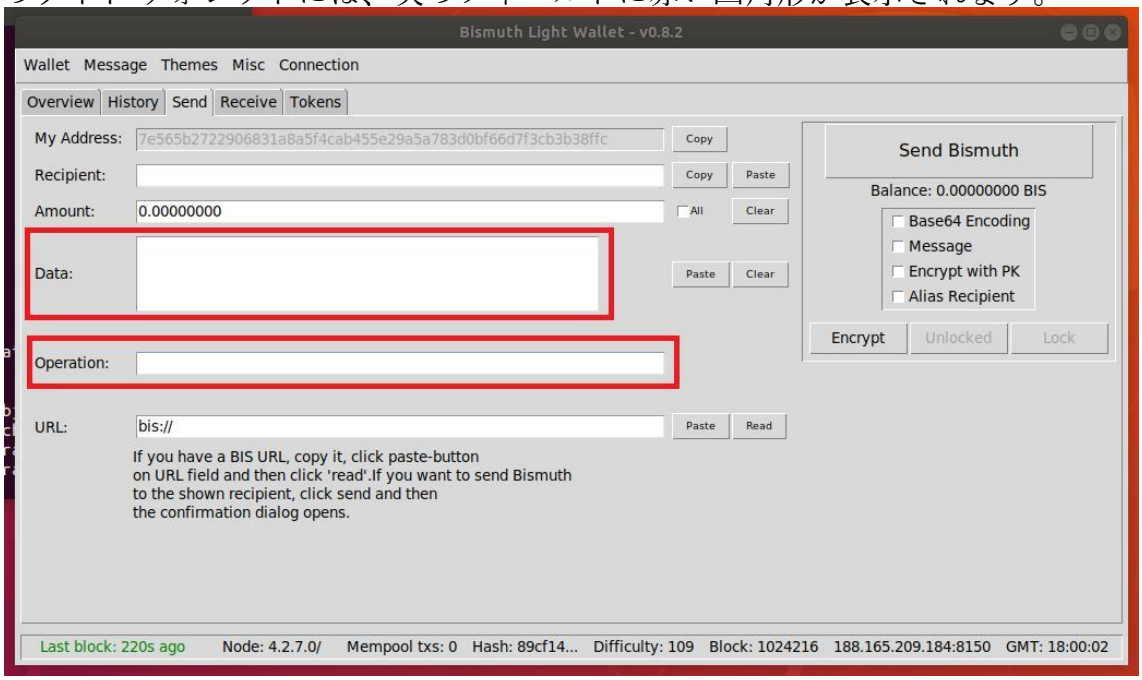
1. トランザクションの処理時間が長い場合は望ましくありません。所望の平均ブロックタイムよりも大きい多くの要因である処理時間は、負と見なされます。
2. ブロックチェーンフィードバック制御アルゴリズムは、通常、長いテールブロックタイムと鉱夫の計算能力が低下した状況を区別することはできません。したがって、長いテール ブロックタイムは、通常、応答の速いコントローラが難易度を下げる原因となり、この動作によってプロセスの不要な振動や不安定さが発生する可能性があります。



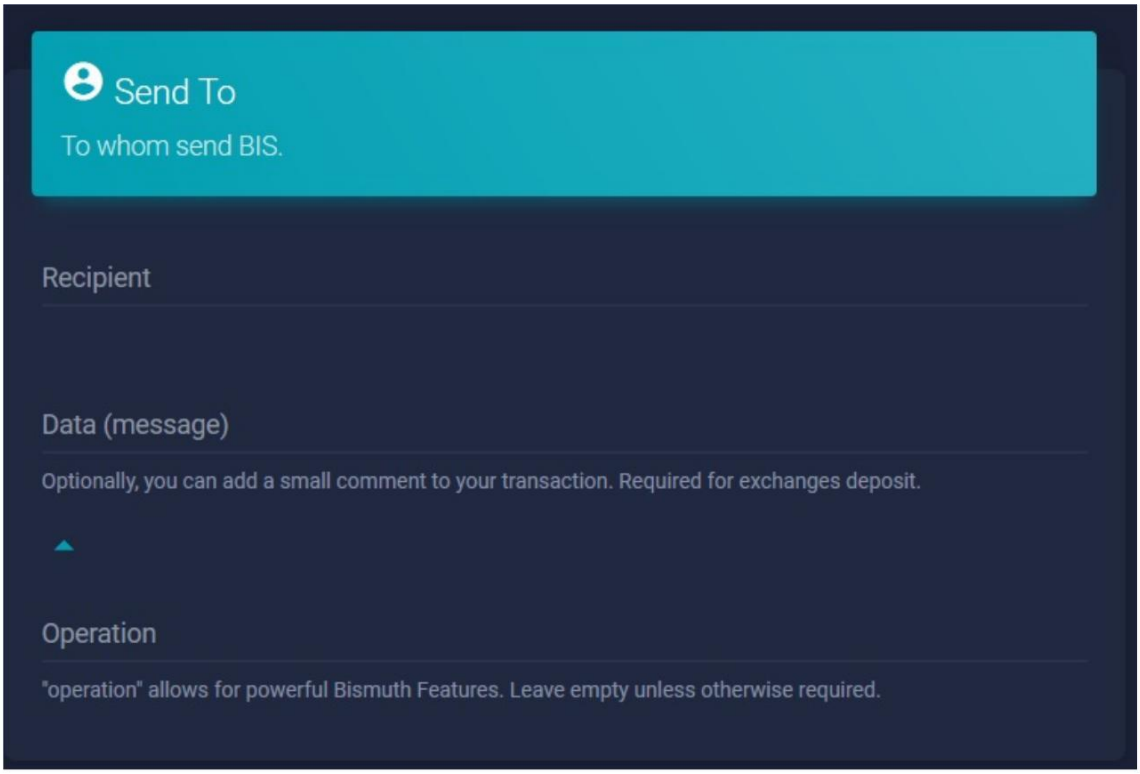
ビスマスでは、長いテールブロックを除去するためのソリューションが実装されており、結果は上記の図に示されています。左側の図は、尾の除去コードがアクティブ(赤い曲線)をアクティブにせず、尾の除去がアクティブ(青いバー)を持たない PDF を示しています。右側の図は同じですが、180 秒を超えるブロックタイムで拡大表示されます。これらの図から見ると、ビスマスでは長いテールブロックタイム(180 秒を超える)のプローブ能力が大幅に低下し、トランザクションのタイムリーな実行が保証されています。実際の実装に関する詳細なインフォーメーションについては、ジャーナル記事:J.クチェラと.ホヴランド、「尾の除去ブロック検証：実装と分析」を参照してください http://dx.doi.org/10.4173/mic.2018.3.1

操作とデータ フィールド

Bismuth には、dApp 開発者が使用できる 2 つのフィールドがあります。次の図のライト ウォレットには、次のフィールドに赤い四角形が表示されます。



2 つのフィールドは、トルネード ウォレットでも使用できます。



このリンクの例を使用するなど、操作フィールドとデータフィールドをプログラムで使用することもできます。このリポジトリのコード例を次に示します。

```
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}")
```

このサンプル コードは、0 BIS を自分自身に送信します。BIS を別のアカウントに送信するには、クライアント アドレスを文字列としてアカウント アドレスに換えます: "9ba0f8ca03439a8b422b256a5f56f4f563f6d6d8355f6d8355f525992fa5daf"

操作とデータフィールドを使用するには、次のコマンド例を使用します。

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

操作とデータフィールドの使用は、他の暗号通貨との大きな違いです。現実世界の原則により、Bismuth プロジェクトは、データをチェーンに格納する必要性を認識し、ユーザーにとって困難ではありません。代わりに、Bismuth はそれを、より高いレベルの操作に役立つメタ データとして使用します。これは純粋なビスマスの精神です:抽象データ、読み取り/書き込みが容易で、ノードは操作/データフィールドが何を意味し、プロセスを知ることなく処理しますが、参加しているアプリは解釈し、行動することができます。操作/データフィールドは、ノード(ベースレイヤー、トランスポート/信頼性/不変/抽象データ)と dApp(データを使用し、解釈し、動作し、より高いレベルの操作)を分離することもできます。

プライベート契約

上記のセクションで説明したように、Bismuth には、余分なプロトコルによってレバーを熟成できる 2 つのフィールドがあります。これらのフィールドは、プライベート コントラクトを作成するさまざまな方法で使用できます。

1. プライベートデータのようにプライベート、すなわち。暗号化されたメッセージ。

2. 非パブリック契約コードのようにプライベート.

3) 抽象トランザクションと暗号化されたメッセージを使用して、プライベートで追跡不可能な受信者のようにプライベート.

抽象トランザクション

BIS の強みの 1 つは、抽象トランザクションを許可することです。これらは、0 BIS が関与するトランザクションであり、そのプロトコルに参加する dApps でのみ理解できるデータです。"操作" フィールドは、"コマンド" 演算子の一種として使用されます。慣例では、名前空間の一種のような簡単な分類を可能にするために、"class:operation" として書式設定された文字列を使用します。オープンフィールドは、関連する (短い) データを保持します。開発者は独自の操作を定義できますが、プロトコルが既に使用されている名前ペースを使用していないことを確認する必要があります。

BIS トランザクション手数料は固定されており、バイト内のオープンフィールドの長さにのみ依存します。料金 = 0.01 + len (オープンフィールド/100000)。オンチェーンストレージは推奨されず、将来的に制限される可能性があります。開発者はペイロードを最小限に抑え、サイドチェーンまたは dApp を使用して実際のデータを格納する必要があります。

ハイパーノードとサイドチェーン

非常に多くの異なるマスターノードコインがリリースされるとき、ビスマスハイパーノードがとても革新的である理由にいくつかのヒントを与えると便利です。

技術ラボとしてのビスマスハイパーノードとビスマス能力の証明

ビスマスが成長し、チームがますます多くの経験を集めるにつれて、ノードのいくつかの弱さがより目に見えるようになります。多くの人や組織で使用されている実行中のブロックチェーンで設計をテストまたは変更することは必ずしも可能ではありませんが、ハイパーノードをラボとして使用して新しいテクノロジ、ライブラリ、アルゴリズムをフィールドテストすることは簡単です。Bismuth ハイパーノードには、後でコア コードで使用できるいくつかの新しいテクノロジ 層が含まれます。

ハイパーノードは、ビスマス抽象トランザクションの統合と活用の容易さも示します。ハイパーノードは、ビスマスが提供するオープン性と抽象化層で何ができるかの良い例です。

ネットワーク値

ビスマス ハイパーノードの目標は、ネットワークに付加価値を提供することです。いくつかの基本的なマスターノードの実装は、単に「ping」の答えを取り付けるか、またはチェックしています。

ビスマス ハイパーノードは、全く異なるレイヤで動作します。代わりに、通貨ではなくメトリック (主要業績評価指標 (KPI)) を使用して、独自にチェーンを使用します。両方のチェーンは疎結合されており、非常に異なるルールで、ほぼ不変の方法で動作します。ハイパーノードは、マイニングもハイパーオデの間の競合もなく、ステーク証明(PoS)チェーンで動作します。KPI を監視し、PoS チェーンに格納します。ハイパーノードは PoW チェーンの一部ではないため、追加の攻撃ベクトルは追加されません。.

適格なハイパーノードは、PoW チェーンに格納されている不変情報から派生します。PoS ノードと PoW ノードの両方からの品質インデックスと不適切な動作が記録され、後でアクションを実行するために PoS チェーンにも変更不能になります。

2 つのチェーンの独立性により、次の点が保証されます。

a/ それは非常に異なる方法で両方のチェーンを偽造する必要があるため、ネットワークを操作することははるかに困難です。

b/ 悪い俳優や不正行為の試みは、独立した不変の方法で記録されます。あなたは気づかれずにカンニングすることはできません。

ビスマスは、ネットワークを監視し、アクターのフェアプレイを保証する統合された非依存型 KPI 専用サイドチェーンを備えた最初の暗号通貨である可能性があります。

ビスマス ハイブリッド PoW/PoS

ビスマス PoW/PoS で使用されるプロトコルは、ハイブリッドな 2 層アプローチです。これは、PoW と PoS の両方からの強さを使用し、他のハイブリッドアプローチの落とし穴を回避しようとします。PoS 層は、主要なコンセンサスの不可欠な部分ではなく、PoW チェーンの公平なオブザーバーとして機能し、そのメトリックは、問題の中核的な原因である悪いアクターに基づいて行動するために使用することができます。

PoS は PoW 俳優を見守り、PoW は PoS 俳優になれる人を決定します。それは、他の 1 つをある程度制御する各チェーン、オロブーロスのようなものです。セキュリティの面では、両方のチェーンが同時に、一貫した方法で攻撃され、いくつかの利点を得る必要があるため、それは大きな改善です。各チェーンのメカニズムは非常に異なっているので、これは壮大なタスクです。

このようなサイドチェーンの将来の使用

ハイパーノードの疎結合の 2 層アプローチには、いくつかのアドバンタージと、多くの将来の用途があります。次のような可能性があります。

1. 逆に、より多くのブロックを鍛造したり、他の人が自分の順番を取るために鍛造するのを妨げるインセンティブはありません。

2. 現時点では、1 つの PoS チェーン、つまりメットリックを持つハイパーノードがあります。しかし、主要な PoW チェーンを過負荷にすることなく、任意の数の PoS チェーンをビスマスネットワークに追加することができます。サイドチェーンとして見ることができ、柔軟性が高くなります。

3. チェーンは疎結合なので、同じプロトコルを他の通貨で使用できます。PoW であるほぼすべての通貨は、ビスマスのようなハイパーノード層を追加し、このハイブリッドアプローチのセキュリティだけでなく、柔軟な側面または子チェーンの恩恵を受けることができる。

そこで、Bismuth は、実用的でフィールドテスト済みのコードを使用して、新しい領域を探求し続けています。目標は、ハイパーノード コードが簡単にするためのフレームワークになることです。

サイドチェーンを実行します。このようなチェーンは、独自のルール、ブロック時間、手数料(または手数料なし)、ストレージを持ち、メインの BIS チェーンによって保護されています。

技術を覗く人もいる

概念実証の簡単な証明から、Bismuth コードはフル機能のノードとクライアント コード ベースに成長しました。現在のネットワークとの互換性を保つためには、いくつかの新しいテクノロジを残す必要があります。ハイパーノードはそれによって制約されないため、最初からより近代的なアプローチを使用できます。

非同期 IO

非同期/ 待つ、コルーチンの使用は、現代の Python の大きな強みです。この al-low は、効率的で読みやすい非同期コードを書き込みます。何百ものプロセスのスレッドを生成し、ロックを処理し、レースのコンディションをデバッグするのが難しい必要がなくなった。1 つのデータベース ファイルへの複数の同時アクセスで戦う必要はありません。これ以上悪名高いグローバル インタプリタロック(GIL)の制限はありません。ハイパー ノード内では、非同期が最大限に使用されます。ハイパーノードのコアは、非同期呼び出し自体を使用してコールバックを使用する竜巻サーバーとクライアントです。これは、ノード自体のパフォーマンス (負荷の下ではほとんどリソースを使用しません) と安全の観点から両方の大きなプラスです。

プロトブフ

は、高速かつ効率的なシリアル化プロトコルです。また、ほぼすべての言語で利用可能なバインディングを持っているので、低レベルの交換形式として選択されました。構造体の詳細なテキストエンコードの代わりに、ハイパーノードはプロトブフを使用します。

長所：

1. 高速、低いオーバーヘッド。
2. パケットのサイズが小さい。
3. より多くの有効性コントロール。
4. クロス OS と言語互換性があります。

暗号プリミティブ
ハッシュ
ハイパーノードは、最新の blake2 暗号プリミティブを使用します。それらは速く、安全で、可変出力長を持つ。

キーと署名

ハイパーノード アドレスは、従来の ECDSA 暗号化曲線を使用します。

チェーンカップリング

2 つのレイヤーはどのように相互作用しますか?PoS(ハイパーノード)はどのように PoW(ノード)を使用しますか?各ハイパーノードは、クラシックビスマス(PoW)ノードと共に実行されます。これは、ピア、コンセンサス、ブロックの高さのように、PoW 元帳とノードステータスにアクセスを読み取りました。

これは読み取り専用アクセスです。ハイパーノードは単なるオブザーバーです。このデータから、ハイパーノードは次の方法を使用できます。

A 有効なハイパーノードの安全で不変の共有リストを取得します。各ラウンドの開始時に、ハイパーノードは、ラウンドジュローを決定するために、彼らの共通のリストを必要とします。このリストは、過去のチェックポイントから、安定した確認を行った PoW チェーンから抽出されます。これは、ハイパーノード所有者の有効な登録で構成されます。この開始リストは、PoS レイヤーによってマニプ遅延することはできません。

B PoW ピア上のメトリックを収集します。ピアが PoW ノードに接続するたびに、バージョン、ブロックの高さ、IP、ping 時間、コンセンサス状態、接続状態など、多くの情報が漏れます。ロールバックなどの特定のアクションについても同じです。接続に失敗したとしても、記録する価値があります。これらは、ハイパーノードが収集し、PoS チェーンに書き込むことができるメトリックです。すべてのハイパーノードによってラウンドに集約されると、これらのメトリックを使用して PoW 参加者の状態を評価し、例えば«よし»および«悪い»ピアのリストをコンパイルすることができます。

PoW(ノード)はどのように PoS(ハイパーノード)を使用しますか?

同じように、PoW レイヤーはハイパーノードによって提供されるデータを使用できます。

1.不良ピアと良好なピアの動的なリスト。

2.現在の正味の高さに関するより信頼性の高い情報。

これにより、ノードは悪い営業担当者のピア (クラウド内の偽のノードの群れ、またはマイナー ノードをターゲットとする偽のノード) や、不良ブロックにスタックしている古いノード、メンテナンスされていない、または古いバージョンで回避できます。したがって、PoW プロセスの一部であるのではなく、複雑さと攻撃ベクトルを追加するだけで、PoS チェーンは PoW と PoS アクターの修飾に使用される公平な余分なデータにアクセスできます。

メトリック - KPI

多くのメトリックを使用して、ここで使用できます。ハイパーノードの役割は、それらをすべて収集するために事前に行われるため、チームはどれを分析し、どちらを使用するか、およびその方法を決定できます。メトリック自体は、時間とともに進化することができます。アクティブなメトリックとその対応するトリガーも可能です。ビスマスの開発者は、これが最初に手作りされ、その後ガバナンスパラメータに進化すると信じています。ハイパーノードおよび/またはノードの所有者は、観察された悪い動作を軽減するために、さまざまなトリガーとレベルに投票できる可能性があります。正確なメトリックとその使用方法については、進化する別のドキュメントで詳しく説明します。

ハイパーノードのペイアウト

通常のマスターノードメカニズムでは、鍛造マスターノードは、鍛造ブロックごとに報酬を受け取ります。これは、あなたがした場合、カンニングするインセンティブがあることを意味します

より多くのブロックを取得したり、他のブロックを拒否、あなたはより高い報酬を期待することができます。ビスマスハイパーノードでは、報酬はあなたが偽造したブロックに直接関連していないため、このような試みはあなたに対して再生されます、 そして、誰もがあなたがカンニングし、PoS 元帳に記録するのを見るでしょう。また、ビスマススキームでは、すべてのハイパーノード所有者が定期的に支払われます。ランダム性は関係ありません。すべての終了ラウンドは、すべてのアクティブなハイパーノードの支払いにつながります。ハイパーノードの支払いは、いくつかの方法で処理できます。最初の手順では、プライベート契約によって処理され、調整が可能になり、チームは安全であることを確認します。

このプロセスは次のようなものです。

1. 期間に与えられた報酬金額があります(固定またはガバナンスパラメータ)。

2. 期間の«アクティブ»ハイパーノードは報酬を共有します。アクティブとは、ハイパーノードが有効なハイパーノードのリストに含まれ、メトリックを収集して送信し、ピアとやり取りし、陪審員であればいくつかのブロックを偽造した可能性があることを意味します。

3. 報酬は担保金額に比例する必要があります。1 つのハイパーノードがあり、担保が別のノードの 2 倍の場合は、その報酬が 2 倍になります。

4. プライベート契約は、すべての報酬を計算し、それらを支払います。

5. すべての計算入力の詳細は公開され、すべてのユーザーが確認するために PoS チェーンに格納されます。

6. アルゴリズムも公開されています。

ガバナンス

KPI とレベルは、一部のアクターに対する禁止やアクションをトリガーするために使用される可能性が高いので、ガバナンス プロセスを要求するのは当然です。しかし、打ち上げ時には、プロジェクトの非常に多くの性質を考えると、これは不可能です。チームは、意味のある KPI とその使用状況の概要を把握するために、分析プロセスを手動で処理します。その後、フィルタが自律的になり、ハイパーノードの所有者がさまざまなパラメータに投票する可能性があります。

ハイパーブロック圧縮

ビスマスは、冗長性と速度の両方にデュアル データベース システムを使用します。標準の完全元帳データベースに加えて、ハイパーブロックは同じデータの圧縮バージョンであり、最後の 15000 ブロックのみが含まれ、前のすべてのトランザクションに対してゼロを超える許容残高の合計のみが含まれます。一部の では、ハイパーブロック残高が不一致検出用の元帳データと比較されます。また、ハイパーブロック データベースはノードの起動時にメモリに読み込まれ、ハード ドライブへのアクセスを利用してシステム負荷を最小限に抑え、データベースの反応速度と可用性を向上させます。

ペナルティシステム

Bismuth コンセンサス システムのすべてのノードは、すべてのコンネクト クライアントの動作を追跡します。ペナルティは、単一のブロックロールバックを介してチェーン切り替えを強制するすべてのノードに適用され、ペナルティの半分は正直で、情報に基がった最長のチェーンブロックを提供するために削除されます。ローカルコンセンサスのために将来的に遠すぎるノードは自動的に禁止されます。この一連のルールでは、攻撃者がコンピューティング能力の半分以上を所有するだけでなく、システム内のすべてのノードに接続し、同時にすべてのノードを攻撃し、大多数のノードをセットアップする必要があります。それでも、攻撃されるたびに攻撃が徐々に難しくなります。

テストネット

Bismuth テストネットは、Bismuth メインネットと同様に、異なる IP アドレスを持つノードのネットワークで構成されていますが、ノード数は少なくなります。テストネット ノードをセットアップするには、ファイル config.txt で次のパラメータを定義する必要があります。

```
port=2829

version=testnet

version_allow=testnet

testnet=True
```

テストネット上のマイニング アルゴリズムの難易度は、意図的に低くなっています。したがって、開発者は簡単にテストネット上で BIS コインを生成するために、ローカルプールとマイナーを設定することができます。このためには、オプティプールウェアをお勧めします。

雨

regnet ノードをセットアップするには、ファイル config.txt で次のパラメーターを定義する必要があります。

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

テストネットまたは regnet が実行されていることをテストするには、次のコマンドを使用できます。

```
python3 commands.py statusget
```

regnet からの出力は次のようになります。

```
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}
```


レグネットの場合、マイナーを設定する必要はありません。現在のウォレットアドレスをマイナーとして持つ「生成」コマンド instamines N ブロック。たとえば、10 ブロックを「生成」し、その後、regnet で送信をテストできる bis を持つことができます。その後、1 ブロックを「生成」し、あなたのメムプールトランザクションがインスタミットされます。

Regnet はブロックの高さ=1 から始まり、ブロックチェーンをネットワークピアと同期する必要がないため便利です。新しい dApp の多くの機能は、regnet の初期段階でテストできます。分散ネットワーク機能をテストする必要がある場合にのみ、開発者は regnet からテストネットに切り替える必要があります。

教育・研究

ビスマスは、教育と研究のための理想的なプラットフォームです。ビスマスメインネットとテストネットを使用して行われた研究の例は、次のジャーナル記事です。

1. J.クチェラと G.ホヴランド、「尾の除去ブロック検証:実装と分析」、http://dx.doi.org/10.4173/mic.2018.3.1

2. G.ホヴランドと J.クチェラ、非線形フィードバック制御と作業証明ブロックチェーンの安定性分析」、http://dx.doi.org/10.4173/mic.2017.4.1

Regnet は、分散ネットワークに参加しなくても、各学生が自分のコンピュータでローカルの regnet を実行できるため、教育現場で特に役立ちます。ブロックチェーン技術の基本的な概念の多くは、ビスマスレグネットを使用して教えることができました。ビスマスを使用して新しい機能や機能を脱ベロップする研究者や学生は、https://discord.gg/4tB3pYJ で、不和に関するビスマスコア開発チームに連絡することをお勧めします。新しい機能と機能は、コアチームによる追加のテストの後、Bismuth ノードコードに組み込む可能性があります。

将来の見通し

Bismuth プロジェクトの目標は、コアノードを十分に文書化し、可能な限り合理化され、効率的に行い、Bismuth の上に構築する将来の拡張機能とアプリケーションをプラグインシステムと 9 時間分散リポジトリを使用するように奨励することです。

Bismuth はすでに機能豊富な暗号通貨であり、このホワイトペーパーでは、既に実装、テスト、および作業中の部分に関する　を使用しています。今後、より多くの機能が開発されるにつれて、その機能は文書化され、このホワイトペーパーの更新版に含まれる予定です。コア チームには、この GitHub リンクで利用できる開発ロードマップがあります。

概要

このホワイトペーパーでは、ビスマス暗号通貨の哲学、柱、特徴の概要を説明します。ビスマスの 3 つの柱は、1)現実世界の原則、つまり、コア開発者チームが新しい機能を導入する際に実用的なアプローチを取ることを意味し、2)保存する必要性、特別なフィールド(操作とデータ)が提供されていることを意味します。開発者は、分散アプリケーションやプライベートスマートコントラクトを組み込み、インポアタントデータをチェーン上に保存し、3)明確な信頼ラインを保存し、ユーザーがサービスのオペレータを信頼する必要があることを意味します。ビスマスの信頼は、プライベートコントラクトを公開することによって奨励され、ユーザーは時間をかけて分割内契約に関連付けられているトランザクションを監視することができます。

ビスマスのコア機能のいくつかは、いくつかの詳細に提示され、議論されています。Bismuth 暗号通貨とその実装に関するより詳細な情報を求める読者のために、リンクとリファレンスが提供されています。

免責 事項

このホワイト ペーパーの情報は情報提供のみを目的としており、投資や財務に関するアドバイスは含まれていません。投資判断の前に、ご自身でご確認ください。このドキュメントの情報はいずれも、提案、オファー、またはその他のソリチ・テュートが、購入、販売、またはその他の投資関連の活動に関与する、または関与することを控える、または、いかなる投資関連の活動にも関与しない、または信頼されるべきである。暗号通貨。暗号通貨投資は、本質的に揮発性と高リスクです。失う余裕があるもの以上に投資しないでください。

ドキュメントのリビジョン履歴

- 2019 年 4 月 3 日: v1.0 のホワイトペーパーがリリースされました