千葉工業大学
修士学位論文

時空間永続証明のための
ブロックチェーン履歴交差法の検討

所属専攻：電気電子情報工学専攻
学籍番号・氏名：1972042　柳原　貴明

指導教員：藤原　明広 准教授

提出日 令和 3 年 3 月 dd 日

概要

最後に書く

目次

1　序論

（仮）Bitcoin を代表とする (パブリックな) ブロックチェーンは，世界中に散在するコアノード同士で，取引データやブロックを転送するため頻繁に共有し合っている．地理的に近いコアノード同士では通信遅延時間の影響は少ないが，遠距離のノードにブロックを転送する場合，数十秒〜数分の通信遅延が起きることが知られている．

この遅延により，Proof of Work（ PoW ）によるブロック生成時間を十分遅くする必要があるこの理由によりビットコインではブロック生成時間が 10 分になるように制御されている．

この遅延時間により，単位時間に処理可能な取引量が少なくなる．これは「スケーラビリティ問題」として知られている．この問題を解決する方法の一つとして，地理的に近いコアノード同士でドメインを自律的に形成し，ドメイン毎に独自のブロックチェーンを管理する方法が提案されている [2][3]．しかし，各ドメインに存在するコアノードの数が減少するため，ブロックチェーンの非中央集権性と改ざん耐性が低下する問題があった．本章では，本研究の背景として現状の動向と先行研究の問題点を挙げ，本研究目的を述べる．

1.1　背景

近年，クラウドコンピューティングおけるデータサーバーの仮想化を代表されるようなストレージやネットワーク，アプリケーションなどの様々な機能が仮想化技術によって提供されている．利用者は、メンテナンスフリーや維持費の安さなど利点は多くあり普及が加速しているがデータサーバーを管理する大きな力を持った企業・組織に依存することになる．

ブロックチェーンので

サトシ・ナカモトが提唱する以前は、分散システムには貨幣や資産のようなデータで扱えない決定的な欠点があった（ブロックチェーンが成し遂げたこと）

ソフトウェアシステムの構造（アーキテクチャ）には大きく「集中システム」と「分散システム」の 2 つがある．図のように集中システムは 1 つの中心があり，そこに他のコンピュータが接続する形になっており，

分散システムは，複数のコンピューター同士が接続し合って形成されます．システムによっては，分散システムと集中システムを組み合わせることもあり，状況に応じて適切な構造を採るように工夫されている．

集中システムと分散システムそれぞれの長所・短所は表ーーーの通りであり以下のようになる．比較すると，この 2 つは正反対の性質を持つ構造だと分かります．

集中システムでは，中心にあるコンピューターが動かなくなるとシステム全体が停止（システムダウン）してしまう「単一障害点」と呼ばれるマシンが存在したり，維持管理にコストが掛かってしまったりする欠点があるもののシステムの柔軟性や一貫性を維持するのが比較的，容易である長所があります．またなりよりもシステムにおけるデータの完全性を維持できる

データの完全性は，データにかけているころ所や間違っていることがない性質の

分散システムはシステムにおける処理を多くのコンピューターで分担するためコストを削減することができ，単一障害点がないことでシステムダウンする可能性がとても低くなります．加えて，計算能力を高めていくことが可能です．

ネットワークに接続している多くのコンピューターの計算能力を利用できるため全体の計算能力が高くなると同時に，接続するコンピュータを増やすことで，その計算能力を徐々に大きくすることも可能です．

今では多くの人々が高性能の PC やスマートフォンなどを持つようになっているため，それらが接続し合って大きな計算能力を実現できるのうになりました．

また低コストかつダウンしないシステムであれば，紙幣資産のような重要なデータを扱うことができるように思えます．

（ブロックチェーンが成し遂げたこと）しかし，分散システムには貨幣や資産のような重要データを扱えない決定的な欠点があります．貨幣や資産のデータがネットワーク上のコンピューターによって異なっていると，どのデータを信じれば良いか分からなくなり，データへの信頼性がなくなってしまいます．

このような状況では，重要なデータを扱うことは出来ません．そのため，貨幣や資産などのデータはこれまで，完全性を維持しやすい集中システムを中心に扱われてきました．

しかし，すでに紹介した通り，集中システムには単一障害点という ” 急所 ” があり情報が一カ所に集まっています．このことは長所であるものの，維持管理するために必要な技術やコストが高くなることや，情報が集中することによるプライバシーの問題を浮き彫りにしてしまいます．

そこでプライバシーの問題が小さくコストを抑え計算能力の高い分散しシステムで，集中システム程の利便性を持つことができれば両者のいいとこ取りができます．そこでブロックチェーン技術です

ブロックチェーン技術は多くのメリットを持つ分散システムで集中システムのようにデータの完全性を維持するための技術として開発されたと見ることができるその意味で

は，分散システムの可能性をさらに拡張するために技術であり，新たな扉を開いた技術として熱い視線を浴びることとなったのです．

1.1.1 ブロックチェーンの構造

ブロックチェーンは「ブロック」と呼ばれるデータを数珠状につないだ形をしており，ブロックを要約したブロックに取り込むことで全体の整合性を取っております．

1.1.2 ブロックチェーンの非中央集権性

ブロックチェーンの中央集権性

1.1.3 ブロックチェーンの改ざん耐性

1.2 本研究の目的

本研究では，独自のブロックチェーンを管理する複数のドメイン間で定期的にブロックチェーンの状態を共有することで，互いのブロックチェーンにヒステリシス署名を書き込み合う履歴交差法を提案する．提案手法について理論的および実験的観点から，その有効性を評価した．まず，提案法による改ざん耐性の向上比率について理論的に評価した．次にドメイン間で自律的に通信を行うプロトコルを設計し，実際に提案法を実行するノードを実装した．その上で，実際に履歴交差法にかかる遅延時間を計測し，その成功率も実験的に評価した．その結果，理論的に改ざん耐性向上比率が高いことと，実験的に成功率が高いことを確認した．

この遅延時間により，単位時間に処理可能な取引量が少なくなる．これは「スケーラビリティ問題」として知られている．この問題を解決する方法の一つとして，地理的に近いコアノード同士でドメインを自律的に形成し，ドメイン毎に独自のブロックチェーンを管理する方法が提案されている [2][3]．しかし，各ドメインに存在するコアノードの数が減少するため，ブロックチェーンの非中央集権性と改ざん耐性が低下する問題があった．そこで本研究では，ドメイン間で定期的にブロックチェーンの状態を共有し，互いのブロックチェーンにヒステリシス署名 [4] を書き込み合う履歴交差法を用いることで，スケーラビリティ問題の解決を検討する．まず，履歴交差法によって改ざん耐性向上比率がどの程度向上するかを理論的に評価した．次に，ドメイン間で自律的に通信を行うプロトコルを設計し，実際に提案法を実行するノードを実装した．その上で，実際に履歴交差法にかかる遅延時間を計測し，その成功率も実験的に評価した．その結果，理論的に改ざん耐性向上比率が高いことと，実験的に成功率が高いことを確認した．

2 関連研究

2.1 PoWaP（Poof of Work at Proximity for a crowdesnsing system）

PoWaP とは地理的に近いノード同士が空間領域（ドメイン）を構成して蜜になるネットワークを構成することで，ドメイン毎に独自のブロックチェーンを管理するネットワークシステム図ーーーに概要図を示す．ドメインに分割して制御することで，ブロックの生成する時間が短縮可能である．また、利用者は近くのノードに近距離無線通信でアクセスして取引等の情報を送ることで、ドメイン（地域）に特化した情報をブロックチェーンに保持させることも可能になる。

2.2 藤原さんのやつ２

ふじはらさんのやつ

2.2.1 BBc-1

BBc-1[5] は，プライベートな取引データを履歴交差により改ざん耐性を持たせて保存する (ブロックチェーンではない) 分散台帳技術である．ノードの実装も行われており，その有効性が実験的にも確認されている．BBc-1 では，取引データにヒステリシス署名を追加する．ヒステリシス署名を取引データに付けて，ドメイン間でデータを共有することで改ざん耐性を高めている．また，PoW による改ざん耐性の向上を行わない為，エネルギーコストは低く済む特徴がある．しかし，改ざん耐性がどの程度向上したかを理論的に評価しにくいことも知られている．

2.3 bloXroute

bloXroute [6] では，P2P ネットワーク上で取引データやブロックの共有速度を早める為に，構造化されたネットワークを (Layer0 と呼んでいる) 上位ネットワークに構築する提案を行っている．

2.4 Atomic Swap

ドメイン間で相手を信用することなく通貨を交換できる技術として，Atomic Swap [7] が知られている．

3　関連技術動向

3.1　集中システムと分散システム

3.1.1　集中システム

集中システム

3.1.2　分散システム

分散システム

3.2　Bitcoin

Bitcoin はデジタルマネーエコシステムの基礎となるコンセプトと技術の集合体である．Bitcoin ネットワークの参加者間で，価値の保有と転送が Bitcoin という名の通貨単位で行われます．ユーザー間の通信は主に Bitcoin プロトコルに基づいて，インターネットを通じて行われる．Bitcoin のプロトコルスタックはオープンソースとして利用可能であり，ノートパソコンからスマートフォンまで，様々なデバイス上で動作する．

ユーザーはネットワークを通じて bitcoin をやり取りすることで，従来の通貨で行うほぼ全てのこと，つまり物品の売買から個人や組織への送金，融資まで行うことが可能になります．bitcoin 自体も売買が可能であり，専門の両替機関で他の通貨とも両替することも可能です．bitcoin は高速かつ安全であり，国境を越えて取引が可能であることから，ある意味でインターネットに使うための最適な通貨ともいえます．伝統的な通貨と異なり bitcoin は完全に仮想的なものです．物理的なコインは存在せず，またデジタルコイン自体が存在するわけでもありません．コインは「送信者から受信者へある一定量の額面を移動させる」という取引 (トランザクション) の中で暗に示されるものです．

Bitcoin のユーザーはトランザクションの所有権を証明する鍵を所有し，そのトランザクションの中に記載された額面を使用したり新しい所有者に送金することができます．

この鍵はそれぞれの利用者の PC 上の電子財布 (ウォレット) に保持されます．この鍵の保有が bitcoin を使用する唯一の条件であり，そのコントロールは各ユーザーにゆだねられています．

Bitcoin は分散された peer-to-peer のシステムであり，何らかの”中央”サーバや管理者が存在するものではありません．

bitcoin は取引の過程で行われる”マイニング”と呼ばれる数学的な解を見つけ出す競争

により新たに生み出されます.

どの (フルプロトコルスタックを動作させている)Bitcoin ネットワーク参加者も，自身のコンピューターリソースを用いて取引の記録処理と検証処理を行うことでマイナーとなることができます.

平均して 10 分に 1 回の頻度で誰かが数学的な解を見つけることで取引が検証され，その解の発見者には bitcoin が新たに与えられます.

つまるところ bitcoin のマイニングとは，中央銀行が行う必要があった通貨の発行と決済の機能を，世界的な競争で代替したものなのです．Bitcoin プロトコルにはマイニングの機能を規定するアルゴリズムが組み込まれています．マイナーが行わなければならないタスクの難易度は，マイナーの数 (または CPU の数) が変動しても平均的に 10 分に 1 回ほど解が見つかるように自動的に調整されます.

またプロトコルでは bitcoin が新たに発行される頻度は 4 年毎に半減されるように規定されており，また bitcoin の総発行量は 2100 万 bitcoin を超えないように規定されています．結果，bitcoin の流通数は容易に予想可能で 2140 年に 2100 万に到達するカーブを描くことになります.

長期的に bitcoin の発行レートが減少していくため，bitcoin 通貨はデフレーション傾向となります.

さらに，予期された通貨発行レートよりも多く通貨が”発行”されることがないため，インフレーション状態になることはありません.

Bitcoin はプロトコル名でもありネットワーク名でもあり，さらには、分散コンピューティングのイノベーションの名称でもある.

通貨としての bitcoin はこのイノベーションの単なる最初の応用であるだけです．開発者の視点から，Bitcoin を通貨のインターネット，つまり分散コンピューティングによって価値やデジタル資産の所有権のセキュアなやりとりを行うためのネットワーク基盤と考えています．Bitcoin は見た目よりも大きな可能性にあふれているのです．この章では主な概念と用語を説明することから始め，必要なソフトウェアを用意し，単純な取引の中で bitcoin を使ってみます．その後の章で Bitcoin が動作する技術的な部分を明らかにし，Bitcoin ネットワークとプロトコルの仕組みを見ていきます.

3.3 分散台帳技術

ブロックチェーン

3.4 ブロックチェーン

ブロックチェーンは 2008 年に「Satoshi Nakamoto」という仮名の人物ないし集団が発表がた Bitcoin の基幹技術として生まれ，現在は主に仮想通貨としての活用を目指す技術として注目を浴びている．

ブロックチェーンは 2008 年，匿名の開発者サトシ・ナカモトにより，デジタル通貨システム「ビットコイン」を実現するための分散タイムスタンプとして提案された．

ビットコインは，管理者のいない環境下で電子的に表現されたコインの制御権の移転，すなわち送金を実現することを目標に設計されている．送金の取引は入金 (入力) と出金 (出力) の間の関係を記述するが，ビットコインでは，未使用の取引出力がコインであるとするいわゆる

UTXO(Unspent TX Output) 構造 (図 -1) を用いて電子コインのディジタルデータ形式を定義している．この構造は，適切な秘密鍵を用いて取引にディジタル署名できる主体だけが送金できることを保証するが，署名の検証に必要な情報 (公開鍵) をデータ構造の中に埋め込み，公開鍵のメッセージダイジェスト (ハッシュ値) をコインの送金の宛先とすることで，まったく関係のない第三者でも公開鍵の正当性を確認でき，取引の形式的な正しさを検証可能にした点で画期的である．

3.5 ビットコインにおけるブロック構造

ビットコインブロックチェーンにおけるブロック構造は以下図のように構成されている。

3.5.1 transaction（トランザクション）

トランザクション

3.6 P2P ネットワーク [peer to peer sysytem]

全てが同じである複数プロセスによって構成される．P2P システムで行われるべき機能は，システムを構成するそれぞれのプロセスによって果たされているといえる．したがって，プロセス間の関係の多くは，対称性がある．言い換えると，ピアツーピアシステムにおいて，各プロセスは，クライアントとして振る舞いながら，同時にサーバとして振る舞っていると言える．このことから，サーバとクライアントと振る舞いながらから生ま

れた造語サーバントと呼ばれる.

3.7 ナカモト・コンセンサス

ナカモト・コンセンサスはブロックチェーンにおける重要な役割を果たす

3.8 Proof of Work (PoW)

difficulty ターゲットを満たす SHA256 のアルゴリズムに対し，解を見つけなければなりません.

3.9 暗号学的ハッシュ関数

暗号の利用方法の 1 つに，メッセージの完全性を確認するための情報であるメッセージダイジェストと呼ばれる固定長のビット列を生成するものがある．任意の長さのメッセージから固定長のビット列を生成する暗号アルゴリズムは，暗号学的ハッシュ関数と呼ばれる.

暗号学的ハッシュ関数は，以下の性質を持つことが特徴である.

- 原像計算困難性
- 第 2 原像計算困難性
- 衝突困難性

原像計算困難性とは，ハッシュ値 h が与えられたとき，$h = hash(m)$ となるような任意のメッセージ m を探すことが困難な性質である.

第 2 原像計算困難性とは，入力 $m1$ が与えられたとき，$hash(m1) = hash(m2)$ となるようなもう 1 つの入力 $m2(m1$ とは異なる入力) を見つけることが困難である性質である.

衝突困難性とは，$hash(m1) = hash(m2)$ となるような 2 つの異なるメッセージ $m1$ と $m2$ の組を探すことが困難である性質である．これらの性質によって，2 つのメッセージに対するダイジェストが同じである場合，メッセージが同一である可能性が高いことを意味する.

また，逆にダイジェストが異なる場合は，メッセージは異なるものであることを意味する.

送信者から送られた文書が，正しく送信者が作成したものであるかは，文書全体が送信者の秘密鍵で暗号化されていれば確実である．しかし文書全体を公開鍵暗号方式で暗号す

ることは効率面から難しいし，またその必要のない場合が多い．そこで，文書全体を暗号化するのではなく，文書の完全性を保証する情報を秘密鍵で暗号化して添付することが多い．

それを電子署名（デジタル署名）という．

文書の完全性を保証する情報としては，メッセージダイジェストが用いられる．メッセージダイジェストは入力文書の大きさのよらない固定長の情報（ハッシュ値）であり，以下の特徴を持つ．

- メッセージからメッセージダイジェストが簡単に計算できる．
- メッセージダイジェストからメッセージを見つけることは出来ない．
- メッセージから同じメッセージダイジェストが生成される別のメッセージを見つけることはできない．
- メッセージを 1 ビットでも変更すれば，メッセージダイジェストは全く異なる値になる．

このような特徴を持つメッセージダイジェストを生成するアルゴリズムとして，$MD5$, $SHA-1$ などのハッシュアルゴリズムが従来から使われてきたが，$MD5$ はすでに破られており，$SHA-1$ についても脆弱性が指摘されている．現在は，$SHA-2(SHA-256)$ あるいはその後継として公募された $SHA-3$ への以降が急がれている．

現在暗号学的ハッシュ関数が注目されている応用として，ブロックチェーン技術がある．

3.10 ビットコインにおけるブロック構造

ビットコインブロックチェーンにおけるブロック構造は以下図のように構成されている。Version ソフトウェア／プロトコルのバージョン番号 Previous Block Hash 親ブロック（1 つ前のブロック）のハッシュ値 Merkle Root ブロック内の全トランザクションに対するマークルツリーのルートハッシュ Timestamp ブロックの生成時刻 Difficulty Target ブロック生成時の proof of work の difficulty Nonce proof of work で用いるカウンタ

3.10.1 transaction（トランザクション）

トランザクション

3.10.2 Previous Block Hash

3.10.3 Merkle Root

3.10.4 Timestamp

3.10.5 Difficulty Target

3.11 ハッシュレート

本研究での，ハッシュレートとは，単位時間あたりに暗号学的ハッシュ関数の計算を行うことができる回数である．この時の改ざん耐性は，マイニングに勝利してブロックを生成するのは，ハッシュレートが最も能力のあるノードである為，ハッシュレートの最大値のノードが基本的にマイニングに勝利する．

3.12 公開鍵暗号方式

公開暗号方式は，

3.13 ブロックチェーンコアノードのドメイン分割

ブロックチェーンコアノードのドメイン分割

3.14 ブロックチェーンにおけるスケラビリティ問題

ブロックチェーンにおけるスケーラビリティ問題は，

3.15 スケラビリティ問題へのアプローチ

スケーラビリティ問題のアプローチとして

3.16 電子署名

通常の電子署名は秘密鍵が漏洩することによる改ざんが可能であり，改ざんされたことの検知も不可能である．

一方，ヒステリシス署名では，電子署名が入れ子構造になって署名がされている為，署名の改ざんを行う為には，改ざんした内容以降の全ての入れ子構造になった署名を改ざんする必要があり，改ざんが著しく困難になる．また，部分的に改ざんされた場合は，入れ

子構造を検証することで，改ざんの検知も可能になる．これは PoW に代わる改ざん耐性技術としてブロックチェーンが注目を集める前から知られており，過去のデータ等を改ざん耐性を持たせて保存することができる．

3.17 ヒステリシス署名

ヒステリシス署名は，過去の署名履歴を取り込みながら署名生成を行う方式である．

ヒステリシス署名 [4] とは，通常の電子署名に連鎖する入れ子構造を持たせることで，その改ざん耐性を向上させる技術である．図 1 にヒステリシス署名の構造を示す．1 つ前のヒステリシス署名 S_{n-1} を暗号学的ハッシュ関数にかけたハッシュ値 $H(S_{n-1})$ と一緒に保存したい m 個のデータ $(B_{D_1}, \cdots, B_{D_m})$ のハッシュ値 $(H(B_{D_1}), \cdots, H(B_{D_m}))$ を追加する．この全体の署名 S_n を更に追加して生成されたものがヒステリシス署名となる．

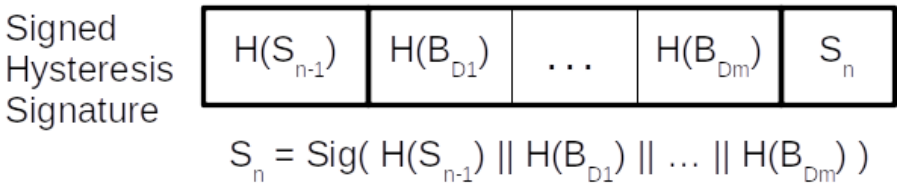

図 1　ヒステリシス署名の例

これにより，有効期限のが切れた電子証明書に対応する電子署名であっても，有効期限内の電子署名書により真正性が保証された電子署名空から連鎖構造を辿ることで，結果として真正性を証明可能となる．

なお通常の署名方式では，署名方式では，署名対象となるメッセージと電子署名とを組みにして相手の（検証者）に送るが，ヒステリシス署名では，それに加えてひとつ前の署名結果のハッシュ値も合わせて送ることが可能になる．

時間情報を文書に織り込むのではなく，署名者が過去に関わった署名付き電子文書の圧縮データを次々に織り込んで行くことである．

3.18 履歴交差法

ヒステリシス署名の証明力はこうしたヒステリシス署名つき文書が作成したエンティティの電子文書が署名を作成したエンティティの手を離れ，他のエンティティの電子文書

におけるヒステリシスとして署名に織り込まれた時に，さらに飛躍的に向上すると考えられる．

それは，第一には，こうした履歴情報の交差によりヒステリシス署名が拡散することじゃ，「過去の電子署名」を偽造しようとするときの作業量を鼠算的に拡大させる空であり，第

4 関連研究

4.1 PoWaP（Poof of Work at Proximity for a crowdesnsing system）

PoWaP とは地理的に近いノード同士が空間領域（ドメイン）を構成して蜜になるネットワークを構成することで，ドメイン毎に独自のブロックチェーンを管理するネットワークシステム図ーーーに概要図を示す．ドメインに分割して制御することで，ブロックの生成する時間が短縮可能である．また、利用者は近くのノードに近距離無線通信でアクセスして取引等の情報を送ることで、ドメイン（地域）に特化した情報をブロックチェーンに保持させることも可能になる。

4.2 藤原さんのやつ２

ふじはらさんのやつ

4.2.1 BBc-1

BBc-1[5] は，プライベートな取引データを履歴交差により改ざん耐性を持たせて保存する (ブロックチェーンではない) 分散台帳技術である．ノードの実装も行われており，その有効性が実験的にも確認されている．BBc-1 では，取引データにヒステリシス署名を追加する．ヒステリシス署名を取引データに付けて，ドメイン間でデータを共有することで改ざん耐性を高めている．また，PoW による改ざん耐性の向上を行わない為，エネルギーコストは低く済む特徴がある．しかし，改ざん耐性がどの程度向上したかを理論的に評価しにくいことも知られている．

4.2.2 bloXroute

bloXroute [6] では，P2P ネットワーク上で取引データやブロックの共有速度を早める為に，構造化されたネットワークを (Layer0 と呼んでいる) 上位ネットワークに構築する提案を行っている．

4.2.3 Atomic Swap

ドメイン間で相手を信用することなく通貨を交換できる技術として，Atomic Swap [7] が知られている．

5　ブロックチェーン履歴交差法

提案するブロックチェーン履歴交差法について説明する.

5.1　提案手法

本研究において提案する P2P ネットワークの構成を図 **??**に示す.

本研究における P2P ネットワークは，Layer0 と 1 の 2 種類から構成される．Layer1 は通常のブロックチェーンを共有する P2P ネットワークである．固有のブロックチェーンを共有している P2P ネットワークに所属するノードの集合のことをドメインと呼ぶ.

今回の提案では，複数のドメインが互いに履歴交差法によりブロックチェーンの「指紋」をヒステリシス署名として共有することで，全ドメインの改ざん耐性での改ざん耐性向上を目指す．Layer1 では，各ドメイン $(D_1, D_2, \cdots, D_m)$ が独自のブロックチェーンを管理する．Layer0 は，Layer1 のドメインを代表する中心コアノード同士が Layer1 とは異なる，Layer0 の P2P ネットワークにより相互接続している．ここでは簡単の為，各ドメインの中心コアノードは 1 つとしているが，一般に複数あっても構わない．このように上位層として Layer0 を付与することは，別の手法でも同様に提案されている [6].

5.1.1　ブロック構造

提案するブロックの構造を図 3 に示す.

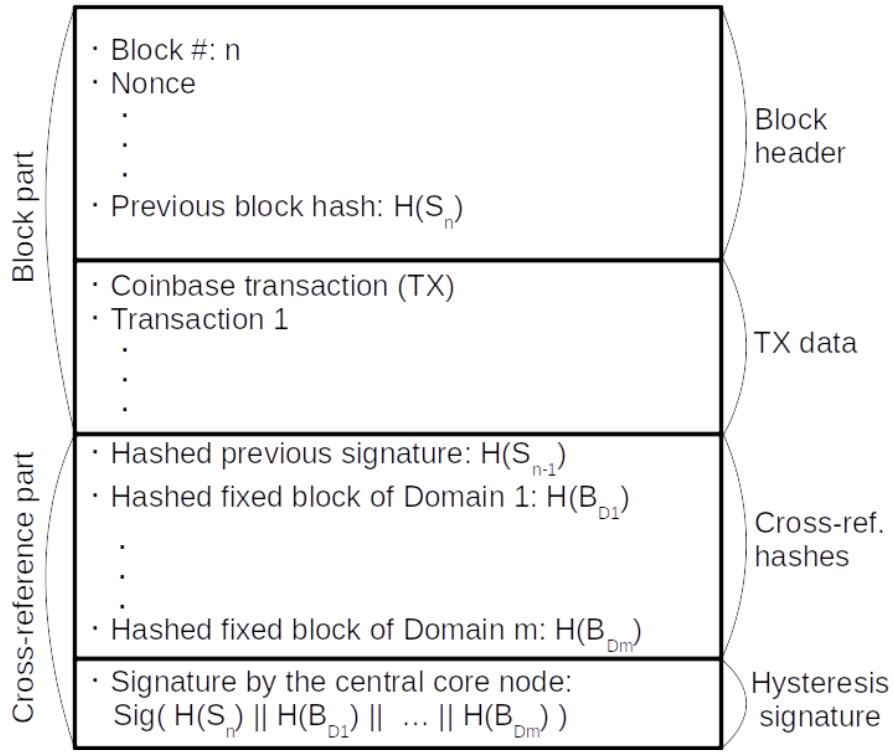

図 3　提案するブロック構造

従来のブロック構造の後ろに履歴交差部 (Cross-reference part) を追加している．履歴交差部に書き込むヒステリシス署名は，Layer0 を介して共有する．

5.1.2　履歴交差法

履歴交差法の動作を図 4 に示す．

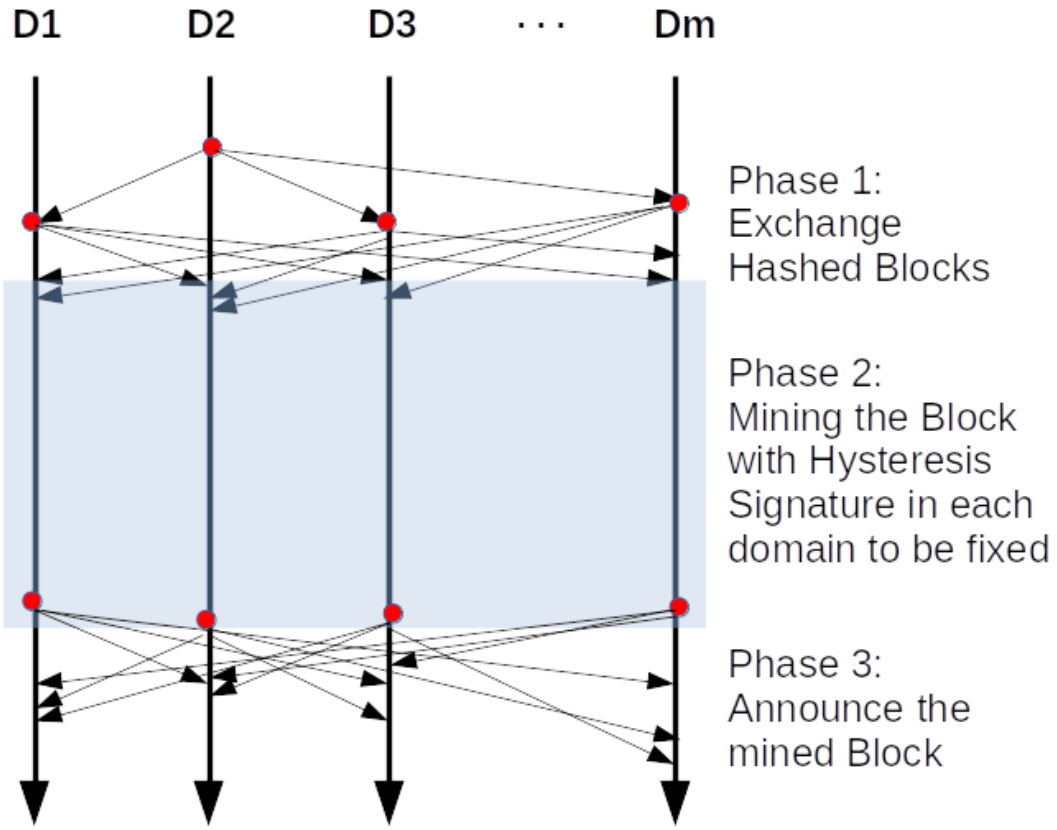

図 4　提案する履歴交差法

全体の動作は，次の 3 つの Phase から構成される．以下にその流れを簡単に説明する．

1. Phase1 では，ドメイン間で l-確定ブロック（最新のブロックから l 個前のブロック）を含むメッセージをドメインを代表する中心コアノード同士で互いに送り合う（ただし l は正の整数）．
2. Phase2 では，各ドメインが届いたメッセージを用いて図 1 のヒステリシス署名を行い，各ドメインが独立に署名を履歴交差部に書き込む為のマイニングを行う．
3. Phase3 では，マイニングし終えたブロックが，l-確定ブロックになったことを他のドメインにブロードキャストして報告する．

5.1.3 履歴交差法の分散アルゴリズム

履歴交差法を実現する分散アルゴリズム [8] を設計した．その詳細を図 5，6 に示す．本研究における履歴交差を実現する 2 つのアルゴリズムが正しく動作する前提条件として，次の 1〜3 が挙げられる．

1. 中心コアノード同士が作る P2PNW は同期システムである．また，その構造は完全グラフとする．
2. 中心コアノードは正しくアルゴリズムを実行し，互いに必ずブロックを転送する．
3. アルゴリズム 1 では，全ての中心コアノードに故障停止がない場合に動作する．
4. アルゴリズム 2 では，t-故障停止がある場合を想定する．つまり，最悪 t 個の中心コアノードが故障停止に陥った場合でも動作する．

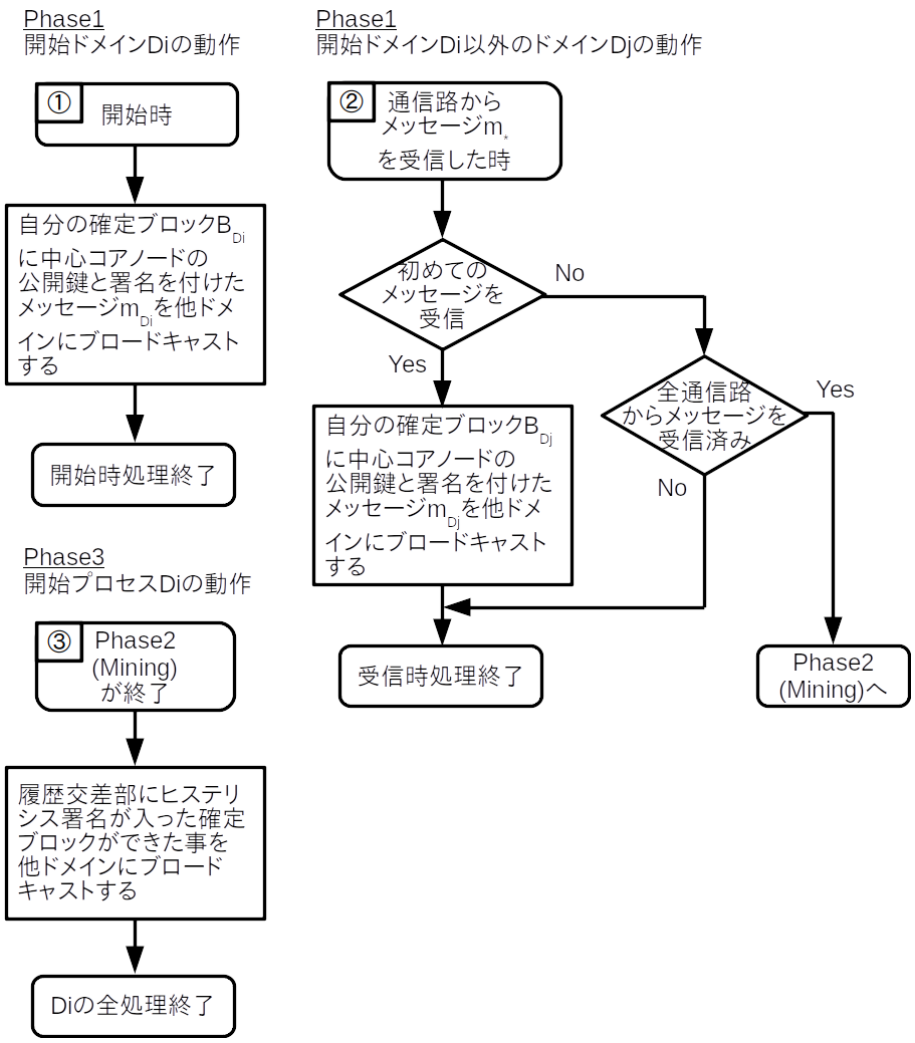

図 5　アルゴリズム 1（Phase 2 は Layer1 における通常のマイニング作業を行うだけである為，省略した.）

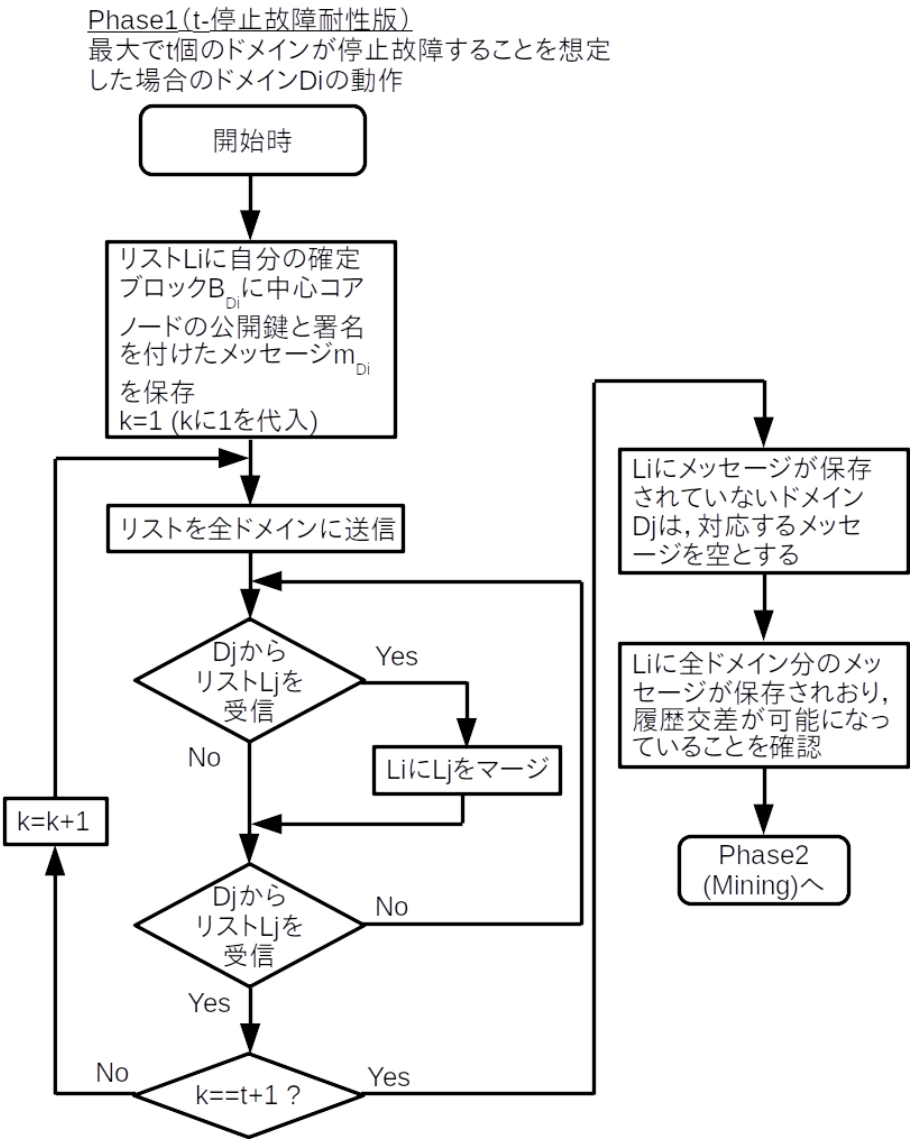

図 6　アルゴリズム 2(Phase 1 のみを記述．その他の Phase はアルゴリズム 1 と共通．)

5.1.4　提案方式における分散アルゴリズムの効率

分散アルゴリズムの効率は，通信計算量と時間計算量で評価することが一般的である．通信計算量のとは，ある問題を解くアルゴリズムが開始してから終了するまでに送信されるメッセージの総数の最大値である．最大値を評価する以外に，平均値での評価やメッセージの総数ではなくメッセージの総ビット数で評価することもある．最大値で評価するのは最悪のケースを想定することを意味する．　非同期な分散システムでは，計算が開始

してから終了するまでの時間を評価することができない．そこで，通信の遅れがプロセス内での計算よりも短いという仮定のもとで，次のような仮想的な同期実行を考える．『第１ラウンドにおいて，各プロセスは同時に実行を開始し，内部計算およびほかのプロセスへの送信を実行し，ほかのプロセスからのメッセージ受信待ちになるまで動作を行う．第 $n+1$ ラウンドでは，第 n ラウンドで送信されたメッセージがすべて受信側プロセスに到着する．各プロセスはそれらのメッセージを受信し，受信したあとに実行する内部計算およびほかのプロセスからのメッセージ受信待ちになるまで動作を行う.』このように動作を行った場合のラウンド数を時間計算量という．時間計算量という．時間計算量においても，最悪値や平均値で評価する．

計算量の評価の例　たとえば，プロセスが P_1，P_2，...，P_n の n 個の存在して各プロセスが直接接続されている（すなわち，グラフ G が完全グラフである）ときに，各プロセス P_i が持つ値 v_i を全員に知らせる問題を考える．ここで，各 vi は b ビットの値とする．単純なアルゴリズムとしては，$A.1$（値の放送アルゴリズム１）のように，各プロセスが全員に放送する，というものが考えられる．

アルゴリズム $A.1$ では，各プロセス P_i は $n-1$ 個のメッセージを送信するので，総メッセージ数は $n(n-1)$ であり，通信計算量は $O(n^2)$ である．また，通信計算量をビットで評価すると，総ビット数は $n(n-1)b$ となり，通信計算量は $O(n^2 \cdot b)$ ビットとなる．

時間計算量については次のようになる．『第１ラウンドで各プロセスが送信を行い，第２ラウンドでそれらの受信を行う．したがって，２ラウンドでアルゴリズムは終了する（ 図 1.19(a) ）.』

また別のアルゴリズムとして各プロセスが P_1 にメッセージを送り，それらをまとめて P_1 から全員に知らせる．という $A.2$（値の放送アルゴリズム２）も考えられる．アルゴリズム $A.2$ では，各プロセス $P_i(i \neq 1)$ からの P_1 への送信が１回，P_1 からの P_i への送信が１回なのでメッセージ数は $2(n-1)$ であり，通信計算量は $O(n)$ である．

また，通信計算量をビットで評価すると総ビット数は $(n-1)b + n(n-1)b$ であり，通信計算量は $O(n^2 \cdot b)$ ビットである．

時間計算量については次のとおりである.『第１ラウンドで各プロセスが Pi に送信を行う．第２ラウンドで P_1 がそれらメッセージを受信し，$P_i(i \neq 1)$ に対してメッセージ送信を行う．第３ラウンドで各プロセスが P_1 からのメッセージを受信する．したがって，３ラウンドでこのアルゴリズムは終了する．（ 図 1.19(b) ）』

アルゴリズム $A.2$ は通信計算量をメッセージ数で評価すると，アルゴリズム $A.1$ よりすぐれているが，通信計算量を総ビット数で評価すると，どちらのアルゴリズムもオーダー的には同じである．また時間計算量についてはラウンド数ではアルゴリズム $A.1$ の

ほうがすぐれている，ということができる．しかし，ラウンド数をオーダーで評価する場合はどちらも定数であるので，オーダー的には同じである．

アルゴリズム 1 が開始してから終了するまでに送信される通信計算量は $O(m^2 \cdot b)$，アルゴリズム 2 の通信計算量は $O((t+1)(m^2 \cdot b))$ となる．また動作を行った場合の時間計算量は，アルゴリズム 1 は，$T_1 + T_2 + T_3$，アルゴリズム 2 は $(t+1)T_1 + T_2 + T_3$ となる．ここで $T_i (i = 1, 2, 3)$ は，アルゴリズム 1 において Phase i にかかる時間である．

5.2 履歴交差法の理論的性能評価

履歴交差法を行うことで，ブロックチェーンの改ざん耐性が向上することが期待される．ここでは，改ざん耐性向上比率を定義し，理論的にその比率を評価する．全ノード数 N で，ノード i のハッシュレートを h_i とする．ここで，ハッシュレートとは，単位時間あたりに暗号学的ハッシュ関数の計算を行うことができる回数である．この時の改ざん耐性は，マイニングに勝利してブロックを生成するのは，ハッシュレートが最も大きなノードである為，ハッシュレートの最大値で与えられる．

$$\max(h_1, h_2, \cdots, h_N) \tag{1}$$

ドメイン数が m 個で，各ドメインに所属するノードの数が D_m 個の場合を考える．各ドメインの改ざん耐性は，ドメインに属するノードのハッシュパワーの最大値で与えられる．

$$A_1 = \max(h_{11}, \cdots, h_{1D1}), \tag{2}$$
$$A_2 = \max(h_{21}, \cdots, h_{2D2}), \tag{3}$$
$$\vdots$$
$$A_m = \max(h_{m1}, \cdots, h_{mDm}) \tag{4}$$

このドメインの中でハッシュレートが最大のドメインが履歴交差法に参加するインセンティブとなるように，全ドメインの中でハッシュパワーが最大のものと比較する．

$$A = \max(A_1, A_2, \cdots, A_m) \tag{5}$$

上記の m 個のドメインで履歴交差法を実行すると，全てのドメインのブロックチェーンを改ざんしない限り，履歴交差部にヒステリシス署名の証拠が残ってしまう為，改ざんが検出されてしまう．従って，改ざん耐性は全ドメインの最大ハッシュレートの和で表される．

$$B = \sum_{i=1}^{m} A_i \tag{6}$$

以上より，ドメイン i が履歴交差法に参加することで，期待される改ざん耐性向上比率 R_i は次式で表される．

$$R_i = \frac{B}{A_i} > 1 \quad (i = 1, \cdots, m) \tag{7}$$

また，ハッシュレートが最大のドメインが履歴交差法に参加することで期待される改ざん耐性向上比率 R は次式で表される．

$$(R_i \geq)R = \frac{B}{A} > 1 \quad (i = 1, \cdots, m) \tag{8}$$

このように，理論的には全てのドメインで改ざん耐性向上率を見積もることができる．

ここで，改ざん耐性向上率 R を見積もる為に，ハッシュレートをランダムに割り振って乱数シミュレーションを行うことにより，数値的に評価する．全ノード数を 1 万とし，ドメイン毎に分割されたノード数を一律 10, 100, 1000 ノードとした場合を評価した．全ノード数を 1000 万ノードとし，ドメイン毎に分割されたノード数を一律 10, 100, 1000 ノードとした場合のハッシュレート h_{ij} （i:ドメイン番号，j:ドメイン内のノード番号）を次式であわらされるパレート分布とした．

$$P(h_{ij}) = \frac{\alpha}{h_{ij}^{1+\alpha}}(h_{ij} > 1). \tag{9}$$

ここで，α はパレート分布のスケールパラメータである．この値が小さいほどハッシュレートに格差が大きくなる．一般にハッシュレートは CPU の数に比例する．CPU をいくつ購入できるかは，ノードを所持する人物ないしはグループの資本と関係している為，富の分布と相関があると予想できる．この理由により，ハッシュレートの分布をパレート分布とすることは妥当であると考える．シミュレーションの実行結果を図 7, 8 に示す．

一般にドメイン数が増加するほど，改ざん耐性向上比率 R の分布のピークが大きい方にシフトすることが分かる．また，ドメイン数が増えると分布の分散も大きくなることが確認できる．また図 7 と図 8 の比較により，スケールパラメータ α が小さくなるほど，改ざん耐性向上比率 R の分布のピークが小さい方にシフトすることも分かる．

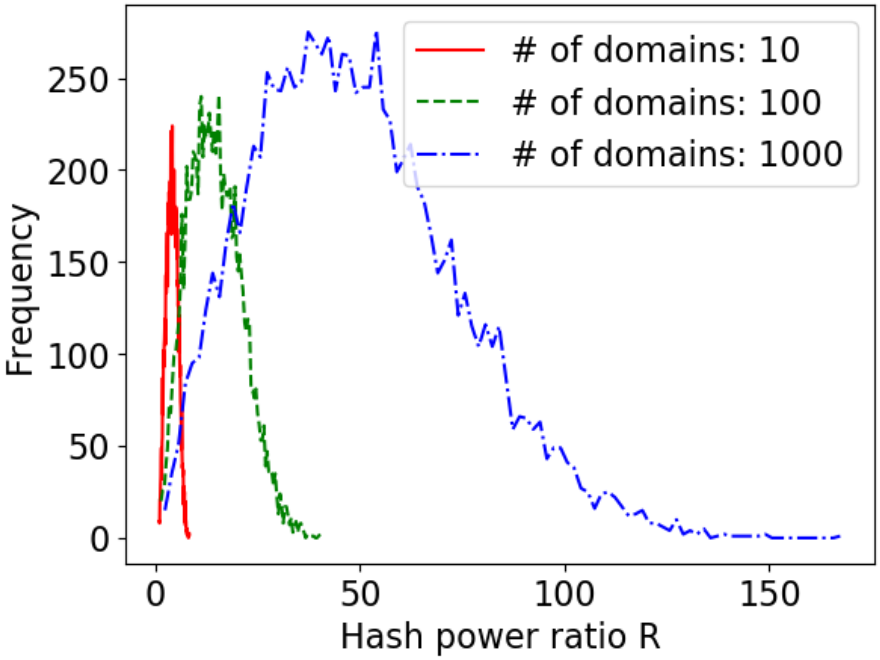

図 7　改ざん耐性向上比率 R の確率分布（ $\alpha = 2$ の場合 ）

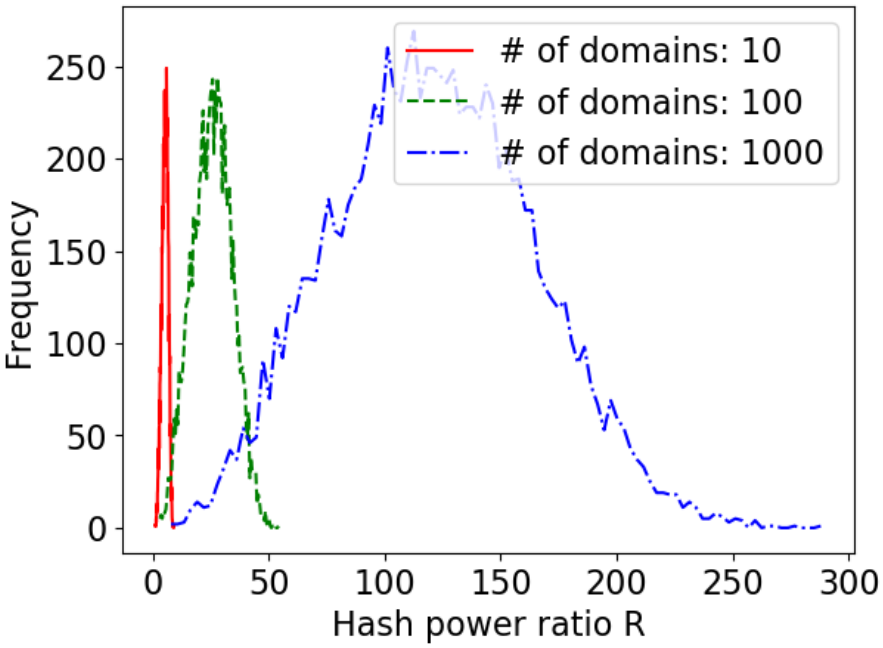

図 8　改ざん耐性向上比率 R の確率分布（ $\alpha = 3$ の場合 ）

5.3 提案するコアノードの実装

提案するおコアノード実装に当たり Layer0 における通信プロトコルの設計を行った.

5.4 通信プロトコルの設計

以下に Socket 通信によって設計した通信プロトコルを示す.

1. **MSG_REQUEST_CROSS_REFERENCE**
   （履歴交差の依頼 ）
2. **MSG_ACCEPT_CROSS_REFFERENCE**
   （履歴交差依頼の承認 ）
3. **MSG_START_CROSS_REFFERENCE**
   （履歴交差の開始号令 ）
4. **MSG_CROSS_REFFERENCE**
   （履歴交差を行いたい block 内容の共有 ）
5. **MSG_COMPLETE_CROSS_REFERENCE**
   （履歴交差の完了通知 ）

6 検証と実装

6.1 実装

6.1.1 通信プロトコル部の実装

5.1 節の提案手法の実験を行う為に，実際に履歴交差法を行うことができるノードを実装した．実装するにあたり，文献 [9] を参考にした．今回は紙面の都合上，アルゴリズム 1 の実装と実験結果のみを紹介する.

以下に Socket 通信によって実装した通信プロトコルを示す.

1. **MSG_REQUEST_CROSS_REFERENCE**
   （履歴交差の依頼 ）

2. **MSG_ACCEPT_CROSS_REFFERENCE**
   （履歴交差依頼の承認 ）
3. **MSG_START_CROSS_REFFERENCE**
   （履歴交差の開始号令 ）
4. **MSG_CROSS_REFFERENCE**
   （履歴交差を行いたい block 内容の共有 ）
5. **MSG_COMPLETE_CROSS_REFERENCE**
   （履歴交差の完了通知 ）

プロトコルを用いて，履歴交差を 1000 回行い，各 Phase が終了するまでにかかる遅延時間分布と履歴交差法の成功率について調査した．ここで，Layer 0 である中心コアノードをドメイン D_1, D_2, D_3, D_4, D_5 からそれぞれ 1 つの 5 ノードからなる状況を想定し，それらの間に P2P ネットワークをローカルに構築した環境において実験を行った．今回は簡単の為に，特定（ドメイン D_2）の中心コアノードが 2 分間隔で履歴交差の依頼を他ドメインの中心コアノードに行い，履歴交差を開始させることとした．

Phase1 では, ドメイン D_2 の中心コアノードが履歴交差の依頼 (**MSG_REQUEST_CROSS_REFE**
を他ドメインの中心コアノードに送信し，ドメイン D_2 以外の中心コアノードが受信する．依頼を受信した各ドメインは履歴交差の依頼を了承する返信 (**MSG_ACCEPT_CROSS_REFFERENCE**) をドメイン D_2 に送信する．ドメイン D_2 の中心コアノードは，他の全てのドメインから了承の返信を受け取ったら，履歴交差を開始する号令 (**MSG_START_CROSS_REFFERENCE**) を他ドメインの中心コアノードに送信し，履歴交差を行いたい l-確定ブロックのデータを各ドメインの中心コアノードにブロードキャストする (**MSG_CROSS_REFFERENCE**)．同様に号令を受け取った D_2 以外の中心コアノードは，各ドメインの中心コアノードにブロードキャストを行う．個々のドメインの中心コアノードは全てのドメインからの確定ブロックのデータを受取り，データを暗号学的ハッシュ関数にかけて hash 値にする．以上で Phase1 は完了となる．

Phase2 では，Phase1 の履歴交差部を書き込む為のマイングを行う．今回の実験では，PoW の難易度を 5 とした．今回実験を行ったマシンでは，この難易度でブロックを生成するのに 10 秒程度の時間がかかっている．履歴交差を行ったブロックが l-確定ブロックとなった時点で Phase2 は完了となる．

最後に Phase3 では，マイニングを完了したブロックが，履歴交差を行ったブロックか

ら l-確定ブロックとなった段階で，他ドメインの中心コアノードに履歴交差に成功したことをブロードキャストする (**MSG_COMPLETE_CROSS_REFERENCE**)．今回の実験では，$l = 3$ とした．ブロードキャストを送信した時点で Phase 3 は完了となる．

6.1.2 通信遅延の考慮

なお，今回の実装には通信遅延を考慮し実装した．各ドメインが Balcelona，Tokyo，Paris，Toronto，Washington にあるとし，履歴交差の依頼を行うドメイン D_2 は Tokyo のものとした．2020 年 2 月 9 日の ping の RTT にかかる時間 [10] を参考にすることで，ポート番号でノードを区別して，sleep 関数を利用することで遅延を発生させた．

参考にした文献 [10] は，unix コマンドラインツールの ping を用いており，ウインドウサイズを今回は 64[Byte] とすると，ping の RTT を P[ms] から，送信データサイズを B[Byte] に応じた通信遅延 L は，以下の式で表される．

$$L = \frac{BP}{64}[sec] \tag{10}$$

送るデータサイズと送り先のポート番号に応じて通信遅延を発生させた．通信する際におおよそ 0.1〜2 秒程度の通信遅延 L が発生している．実際に参照した遅延時間を図**??**，算出した Tokyo から 150〜500[Byte] のデータサイズを他のドメインへの送信した際の通信遅延 L の上限値を表 1 に示す．

表 1　Tokyo からの各ドメインへの通信遅延 (小数第 2 以下は省略)

[sec]	Barcelona	Paris	Toronto	Washington
Tokyo	0.5〜1.9	0.5〜1.9	0.3〜1.2	0.4〜1.3

7　まとめと今後の課題

7.0.1　結果

7.0.2　理論的性能評価

7.0.3　実装システム

前節で実装したプロトコルを用いて，履歴交差を 1000 回行い，各 Phase が終了するまでにかかる遅延時間分布と履歴交差法の成功率について調査した．前節で解説した通信プロトコルを用いて実験を行った結果として，各 Phase での遅延時間の分布を図 10, 11, 12 に示す．

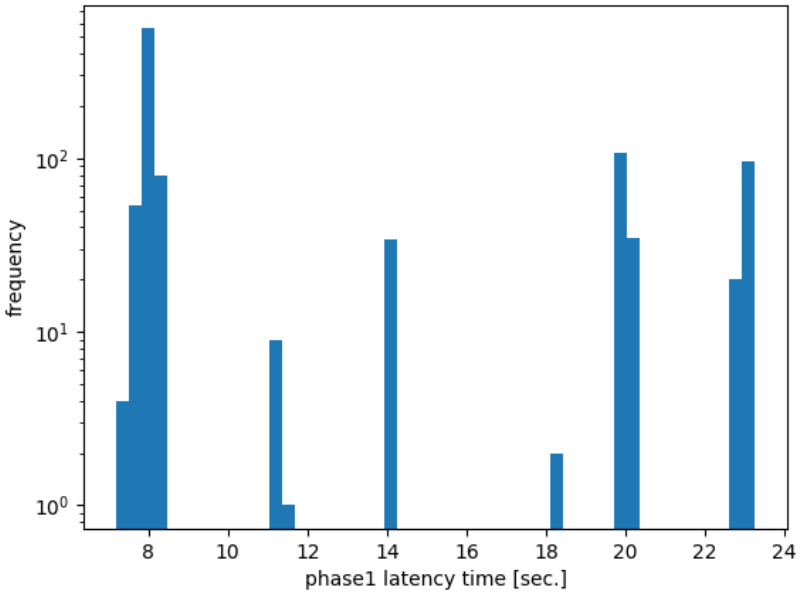

図 10　Phase1 の計測時間

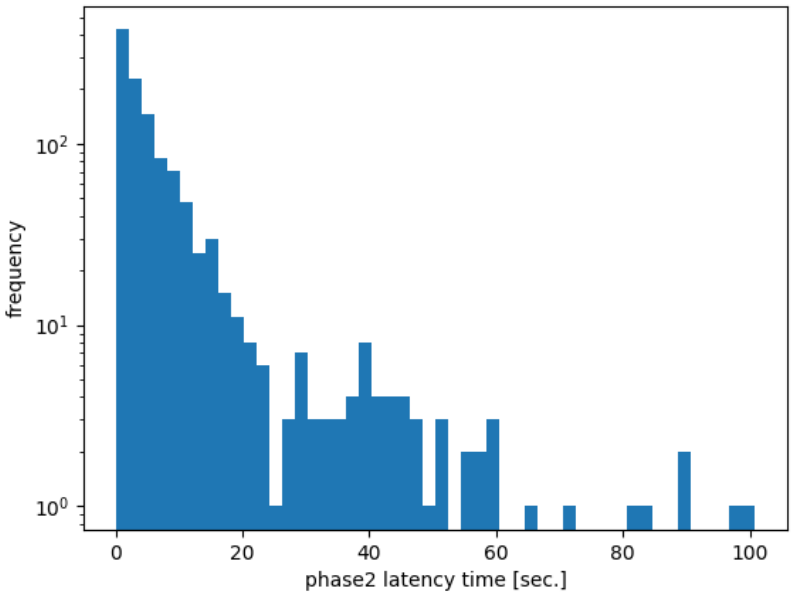

図 11　Phase2 の計測時間

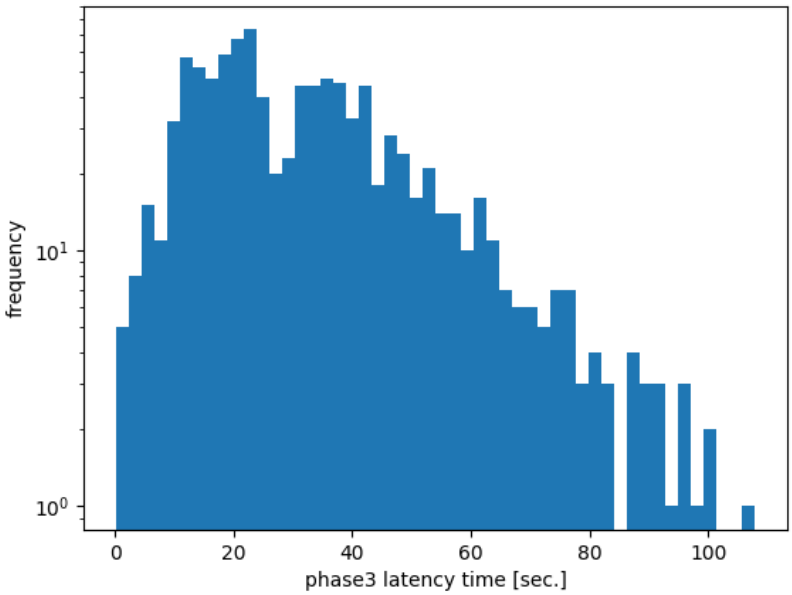

図 12　Phase3 の計測時間

Phase 1 は殆どが 8 秒以下で終わっているが，場合によっては最悪 20 秒程度かかることもあることが分かる．Phase 2 はマイニングを行うことから，指数分布に従っていることが分かる．Phase 3 は全てのノードのマイニングが完了するのを待つことから，タイミングが悪い場合は 100 秒程度待つ必要があることが分かる．全ての Phase を合計すると 120 秒程度になることから，履歴交差にかかる時間を見積もることができた．

また，1000 回の履歴交差のうち，失敗は 0 回であった．従って，成功率は 100% となった．

プロトコルを用いて，履歴交差を 1000 回行い，各 Phase が終了するまでにかかる遅延時間分布と履歴交差法の成功率について調査した．ここで，Layer 0 である中心コアノードをドメイン D_1, D_2, D_3, D_4, D_5 からそれぞれ 1 つの 5 ノードからなる状況を想定し，それらの間に P2P ネットワークをローカルに構築した環境において実験を行った．今回は簡単の為に，特定（ドメイン D_2）の中心コアノードが 2 分間隔で履歴交差の依頼を他ドメインの中心コアノードに行い，履歴交差を開始させることとした．

Phase1 では, ドメイン D_2 の中心コアノードが履歴交差の依頼 (**MSG_REQUEST_CROSS_REFE**
を他ドメインの中心コアノードに送信し，ドメイン D_2 以外の中心コアノードが受信する．依頼を受信した各ドメインは履歴交差の依頼を了承する返信 (**MSG_ACCEPT_CROSS_REFFERENCE**) をドメイン D_2 に送信する．ドメイン D_2 の中心コアノードは，他の全てのドメインから了承の返信を受け取ったら，履歴交差を開始する号令 (**MSG_START_CROSS_REFFERENCE**) を他ドメインの中心コアノードに送信し，履歴交差を行いたい l-確定ブロックのデータを各ドメインの中心コアノードにブロードキャストする (**MSG_CROSS_REFFERENCE**)．同様に号令を受け取った D_2 以外の中心コアノードは，各ドメインの中心コアノードにブロードキャストを行う．個々のドメインの中心コアノードは全てのドメインからの確定ブロックのデータを受取り，データを暗号学的ハッシュ関数にかけて hash 値にする．以上で Phase1 は完了となる．

Phase2 では，Phase1 の履歴交差部を書き込む為のマイングを行う．今回の実験では，PoW の難易度を 5 とした．今回実験を行ったマシンでは，この難易度でブロックを生成するのに 10 秒程度の時間がかかっている．履歴交差を行ったブロックが l-確定ブロックとなった時点で Phase2 は完了となる．

最後に Phase3 では，マイニングを完了したブロックが，履歴交差を行ったブロックから l-確定ブロックとなった段階で，他ドメインの中心コアノードに履歴交差に成功したことをブロードキャストする (**MSG_COMPLETE_CROSS_REFERENCE**)．今回の実験では，$l = 3$ とした．ブロードキャストを送信した時点で Phase 3 は完了となる．

7.1 評価

7.2 考察

8 まとめ

本研究では，独自のブロックチェーンを管理する複数のドメイン間で定期的にブロックチェーンの状態を共有することで，互いのブロックチェーンにヒステリシス署名を書き込み合う履歴交差法を提案した．提案手法について理論的および実験的観点から，その有効性を評価した．改ざん耐性の向上比率について理論的に評価した結果，一般にドメイン数が増加するほど，改ざん耐性向上比率 R の分布のピークが大きい方にシフトすることが分かった．また，ドメイン数が増えると分布の分散も大きくなることが確認できる．スケールパラメータ $\alpha = 2, 3$ の比較により，α が小さくなるほど，改ざん耐性向上比率 R の分布のピークが小さい方にシフトすることも分かった．

次にドメイン間で自律的に通信を行うプロトコルを設計し，実際に提案法を実行するノードを実装した．実際に履歴交差法にかかる遅延時間を計測し，その成功率も実験的に評価した．その結果，120 秒程度待つことで履歴交差法の全 Phase が完了することが見積もれた．また今回の実験では，履歴交差法の成功率は 100% となった．

結論．今後の課題としては，想定される様々な状況においても安定して履歴交差法が行えるように実験を継続することがある．例えば，履歴交差を依頼する中心コアノードを固定しない場合，中心コアノードが変化する場合，履歴交差の依頼が重複してしまった場合への対処，停止故障を考慮した実装（アルゴリズム 2）などが挙げられる．また，理論的に得られた改ざん耐性向上比率を実験的に評価することもある．

謝辞

本研究は，令和 2 年度修士研究として，千葉工業大学大学院工学研究科電気電子情報工学専攻准教授 藤原明広先生のもとで行ったものである．本研究を遂行するあたり，適切なご指導，御助言をして頂いた藤原明広先生に深く感謝の意を表する．同専攻教授　＃ーー先生，並びに同専攻教授　＃ーー先生には本論文を査読していただき，有益なご意見，コメントを頂いた．ここに感謝の意を表する．また藤原研究室の各位には，研究遂行にあたり日頃より有益なご討論ご助言を戴いた．この機会に皆様に深く感謝の意を表す．

参考文献

[1] S.Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System” (2008). https://bitcoin.org/bitcoin.pdf （2020 年 1 月 7 日閲覧確認）

[2] A. Fujihara, “Proposing a system for collaborative traffic information gathering and sharing incentiviz by blockchain technology,” Advances in Intelligent Networking and Collaborative Systems, pp.170-182, Springer (2019).

[3] A. Fujihara, “PoWaP: Proof of Work at Proximity for a crowdsensing system for collaborative traffic information gathering,” Internet of Things, 100046, Elsevier (2019).

[4] 洲崎誠一，松本勉，電子署名アリバイ実現機構 -ヒステリシス署名と履歴交差- 情報処理学会論文誌，Vol.43, No.8, pp.2381-2393, (2002).

[5] K. Saito, T. Kubo, ”BBc-1: Beyond Blockchain One” (2018). https://beyond-blockchain.org/public/bbc1-design-paper.pdf （2020 年 1 月 7 日閲覧確認）

[6] bloXroute, https://bloxroute.com （2020 年 1 月 7 日閲覧確認）

[7] Atomic swap, https://en.bitcoin.it/wiki/Atomic_swap#Algorithm（2020 年 2 月 13 日閲覧確認）

[8] 真鍋義文, 「分散処理システム」森北出版 (2013).

[9] 濱津誠, 「ゼロから創る暗号通貨」PEAKS 出版 (2018).

[10] WonderNetwork, https://wondernetwork.com/pings（2020 年 2 月 10 日閲覧確認）

第 I 部

name

```
#include <iostream>
using namespace std;
int main() {
for(int i = 1; i <= 5; i++) {
cout << "こんにちは，C++ の世界！ " << i << endl;
```

```
  }
  return 0;
  }
```

\usepackage{ascmac}して screen 環境を使うと，枠がつきます。