❸ ビットコイン改善提案　最前線

日向理彦（合同会社ジャノム）
今井崇也（フロンティアパートナーズ合同会社）

ビットコインの技術的進展とビットコイン改善提案（BIP）

非中央集権的な電子マネーを目指して開発が行われているビットコインは，その開発体制自体も非中央集権的になる（特定の開発者が強い決定権を持たない）よう配慮がなされています．そのためには，後方互換性をなくすような新しい機能の追加や，標準的なユースケースの提案に関しては，なるべくオープンな場で公開し誰もが議論に参加できるようにすることはきわめて重要です．ビットコインでは，新しい提案を文書化し誰もがその詳細を自由に閲覧できる場として「ビットコイン改善提案（Bitcoin Improvement Proposal：BIP）」と呼ばれる GitHub 上のリポジトリ（**図 -1**）が用意されています．これはいわば「ビットコイン版の RFC」と言えるような存在であり，新たに取り入れられる機能は必ずこの場で徹底的に議論され，コミュニティ全体の合意が得られた時点で初めて本体のコードに統合される，という運営体制となっています．

BIP に掲載されているもののうち，重要かつ分かりやすいものをいくつか列挙すると，ビットコインアドレスのリンクフォーマットを定める BIP 21，電子署名のシリアライズ方式を厳格化する BIP 66，ピア間通信の暗号化方法を定める BIP 151 等があります．このようにビットコインの重要な仕様に関するドキュメントが多数掲載されていますので，これからビットコインを学ぶ方や現在学んでいる方，また将来どのような機能が実装されていく予定なのか知りたい方にとっては非常に貴重な資料となっております．

そこで本稿では，現時点で BIP にて議論されている新機能のうち，とりわけインパクトが大きく今後

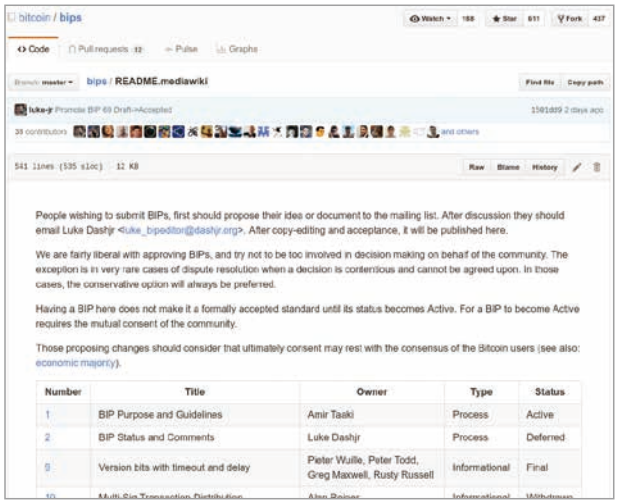

図 -1　GitHub 上の BIP 用リポジトリに掲載されている，現在提出されている BIP の一覧（抜粋）

重要になると思われるものを 2 つピックアップし，それぞれ技術的な内容や意義について簡単に紹介をさせていただきます．

BIP 114：マークル化抽象構文木（Merkelized Abstract Syntax Tree）

ビットコインの仕組みではほかのノードが不正をしていないことを確認するためにネットワークに流れるすべてのトランザクションを検証しなければならないため，ユーザ数が増えトランザクションの作られる数が増えていくとディスク容量やネットワーク帯域が逼迫し耐え切れなくなるのではないかと言われています．この問題はビットコインコミュニティ内では「スケーラビリティ問題」という名前で呼ばれており，長年に渡り優先度の高い懸案事項となっております．当然，これに対する改善策はいくつも提案されていますが，こうした改善策の 1 つとして提案されているのが本章で解説する「マークル化抽象構文木（Merkelized Abstract Syntax Tree：

MAST）」[1] であり，送金したコインの使用可否条件に複雑な分岐処理が含まれている場合に劇的にトランザクションのデータサイズを削減できる仕組みとなっております．

マークル化抽象構文木を理解するためには，送金したコインを誰が利用できるのかをどのように記述しているのかについて抑えておく必要があります．コインの使用可否条件は匿名関数の形でトランザクション内に書き込まれており，たとえばアドレス A（公開鍵 A のハッシュ値）に対する入金トランザクションであれば引数として公開鍵と電子署名を取る，以下のような関数を記述します．

1. 公開鍵のハッシュ値がアドレス A と一致しなければ false を返却し，一致すれば次の処理へ進む．
2. 電子署名が正しければ true を，不正であれば false を返却する．

この関数の実行結果が true なのか false なのかによりコインが出金できるかどうかが決定されます．

P2SH と呼ばれる方式ではこの関数の実データは出金トランザクションに含めるという約束となっており，入金トランザクションには関数データのハッシュ値のみを書き込むことで後から関数が改ざんできないようにしています．

冒頭で軽く触れましたが，マークル化抽象構文木が威力を発揮するのは関数内に分岐処理がある場合となります．例として次のような関数を考えてみましょう．

1. 引数 1 が 0 なら 2. の処理を，1 なら 3. の処理を行う．
2. 引数 2 がアドレス A に対する正しい電子署名になっていれば true，そうでなければ false を返却する．
3. 引数 2 がアドレス B に対する正しい電子署名になっていれば true，そうでなければ false を返却する．

引数 1 を適切に選択することで A さんまたは B さんのどちらでも利用できるトランザクションとなっていますので，これはいわば「A さんと B さんの共用ウォレット」を作っていると言うことができます．

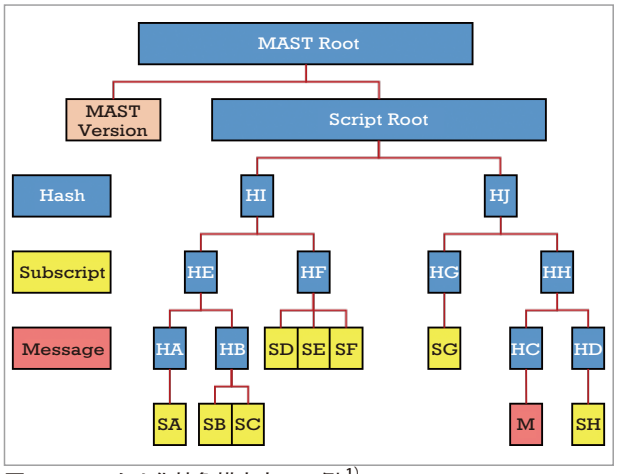

図 -2　マークル化抽象構文木の一例 [1]

ここで A さんがこのコインを出金することを考えると，引数 1 として「0」を渡すことで 1. および 2. の処理を実行することになりますから，関数のうち 3. の処理は一切実行されません．そうすると 3. の処理に相当するデータは落としてしまってもよさそうですが，単純にデータを落としてしまうと関数全体のハッシュ値が正しいかどうかを検証できず，改ざん検知ができなくなってしまいます．そこで分岐先の処理それぞれのハッシュ値を計算し，さらにそれら 2 つのハッシュ値をつなげたものに対してハッシュ値を計算することでこれを関数全体のハッシュ値としてみましょう（マークル木）．こうすることで改ざん検知をするためには，実行されない分岐処理に関しては処理の実データを渡すのではなくそのハッシュ値を渡せば十分だと分かります．これにより分岐先の処理がどんなに長くとも，ハッシュ値の長さ（32 バイト）程度に圧縮することができます．ただし，関数内の分岐が入れ子構造となっている場合などにはこのような単純な構成方法はできませんが，このような場合でも抽象構文木のようなデータ構造を作ることで任意の関数に対応できます（**図 -2**）．このようにして作られた木構造のことを「マークル化抽象構文木」といいます．

マークル化抽象構文木が利用できそうな複雑な分岐処理を含む例としてライトニング・ネットワークに用いられているトランザクションなどが挙げられますが，それ以外に 1 つ興味深い例として「巨大

なマルチシグトランザクション」というものがあります．ここで一般的に M-of-N マルチシグトランザクションというのは，N 個の公開鍵のうちいずれか M 個の電子署名が与えられれば送金ができるという種類のトランザクションのことを指します．たとえば 3-of-2,000 マルチシグトランザクションを考えると，素直な実装では 2,000 個（40,000 バイト）もの巨大なアドレスデータを含まなければいけません．ところが，これを約 13 億個の 3-of-3 マルチシグトランザクションに分解しマークル化抽象構文木を適用することで 1,500 バイト以下のサイズで表現することができます．

ビットコインに対する改善案として利用が検討されているマークル化抽象構文木の仕組みは，抽象的で汎用性が高くほかの分野にも応用できる可能性を秘めています．たとえば安全ではないネットワーク経由で巨大なプログラムの一部分の機能のみダウンロードし利用したい場合に，プログラム全体をダウンロードすることなく改ざんを検知するといった利用例が考えられるでしょう．ビットコインの仕組みの一部として発明されたブロックチェーンに対してほかの分野での応用可能性が模索されているように，マークル化抽象構文木についてもビットコインを超えた分野で活躍する日が来るのかもしれません．

BIP 152：コンパクトブロックリレー（Compact Block Relay）

本章では，2016 年 8 月 24 日（日本時間）にリリースされた Bitcoin Core 0.13.0 から入った新機能である BIP152 コンパクトブロックリレー（Compact Block Relay）[2] を紹介します．Bitcoin Core 0.13.0 は執筆現在の最新リリースになっています．

コンパクトブロックリレーを簡単に説明すると，ノードがブロックを受け取る方法を効率化することでブロックを処理する時間を短縮し，またビットコインの P2P ネットワーク内での通信量を減らそうというものです．これ自体ビットコインユーザにすぐに好影響があるものではありません．この BIP の目的が，ブロックの P2P ネットワーク全体への伝搬時間の短縮にあるわけではありません．しかし，結果的にブロックの伝搬時間の短縮につながり，若干トランザクションの承認が早くなるとは思いますが，劇的に改善するものではありません．むしろ，今後増えていくであろうトランザクションに対して今のうちに無駄を省きネットワークの帯域確保をする対策になっています．

これまでのノード間のブロックのやりとりは，次の通りです．フルノード A の先端ブロックがフルノード B にはまだないという状況を考えましょう（フルノードとは，最初からの完全なブロックチェーンを保持しているノード．簡略化のためフルノードのみに限定）．

フルノード B がフルノード A に getblocks メッセージを送ると，返答としてフルノード A は先端ブロックのブロックヘッダハッシュを inv メッセージでフルノード B に送ります．

次に，フルノード B はフルノード A に先端ブロックのブロックヘッダハッシュを含む getdata メッセージを送ると，返答としてフルノード A は先端ブロックをフルノード B に送ります．

先端ブロックを送るときに，先端ブロックに含まれているトランザクションも一緒に送ります．

さて，最後に送られたトランザクションは，フルノード B が初めて知るものなのでしょうか？

いえ，このトランザクションはもっと前にフルノード B に到達していることが多いのです．トランザクションを発行してほかのノードにブロードキャストすると，P2P ネットワーク上をトランザクションが伝搬していきます．通信速度が遅いネットワークを除いて，数秒で P2P ネットワーク全体にトランザクションが行き渡ります．このため，マイニングをしてブロックが生成されるときにはすでに，ブロックに含まれるトランザクションは P2P ネットワーク上に行き渡っているのです．

コンパクトブロックリレーでは，トランザクションを指し示す ID（short transaction id）だけをもらうことで，いちいちトランザクションを送っても

らわなくてもブロックを再構成でき，ゆえにP2Pネットワーク内の通信量を減らせることになります．

コンパクトブロックリレーの仕様では，コンパクトブロックを利用するときに2つのモードが用意されています．高帯域幅リレー（High Bandwidth Relaying）と低帯域幅リレー（Low Bandwidth Relaying）です（**図-3**参照）．

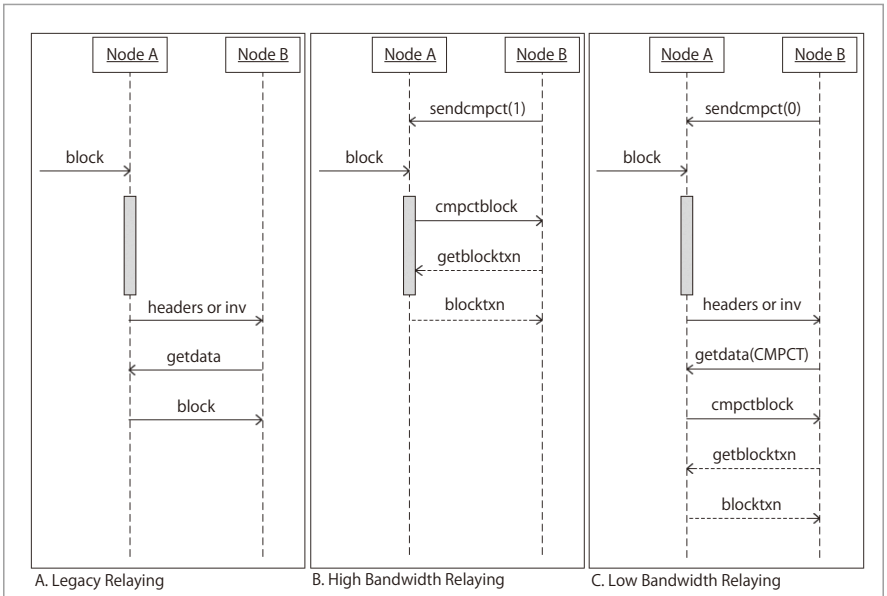

図-3　コンパクトブロックリレーのプロトコルフロー

この2つのモードの大きな違いは，新しいブロックが来たことを次のノードにアナウンスするタイミングです．高帯域幅リレーでは新しいブロックの検証をしながら次のノードにアナウンス（cmpctblockメッセージ）をしますが，低帯域幅リレーではこの検証が終わってからアナウンスをします．低帯域幅ネットワークでは，高帯域幅ネットワークと違ってトランザクションの省略ができない場合（ノードにまだトランザクションが伝搬していない）が多いと考えられます．short transaction idによるハッシュ衝突を避けるため，低帯域幅リレーでは始めにinvメッセージまたはheadersメッセージを使って新しいブロックのアナウンスをすることになります．

詳細な技術内容を知るには

本稿ではコインの使用可否条件が複雑な場合に劇的にトランザクションのサイズを削減することのできる技術「マークル化抽象構文木」および，ピア間でブロックデータを受け渡す際に通信相手がブロックに含まれているトランザクションをすでに持っていた場合に通信量を削減する「コンパクトブロックリレー」を取り上げました．紙面の都合上，具体的なプロトコルなどの詳細な技術情報については掲載せず，概要を理解できる程度の説明にとどめましたので，より詳しく知りたい方は以下に掲載してあります文献2）をご参照ください．なお，本稿に記載しました情報は2016年8月時点のものであり，将来的に変更される場合がありますので，最新の情報につきましてはご自身で改めてご確認いただければと思います．

参考文献

1）Lau, J. : BIP 114 : Merkelized Abstract Syntax Tree（https://github.com/bitcoin/bips/blob/master/bip-0114.mediawiki）
2）Corallo, M. : BIP 152 : Compact Block Relay（https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki）

（2016年9月1日受付）

日向理彦　mhyuga@janom.co.jp

東京大学理学部物理学科を卒業後，長距離相互作用のある系の熱統計力学的性質に関する研究で修士号を取得．前後してビットコインと出会い，その斬新さに魅了される．現在は分散型電子マネーに関する基礎研究の傍ら，決済サービスや取引所の開発・運営を行っている．合同会社ジャノム代表CEO．

今井崇也　takaya.imai@frontier-ptnrs.com

フロンティアパートナーズ合同会社代表CEO．（株）ブロックチェーンハブ技術アドバイザー．一般財団法人ブロックチェーン技能認定協会アドバイザー．新潟大学大学院自然科学研究科修了．理学博士（素粒子理論物理学）．