ループリング (LOOPRING): 分散型トークン取引場プロトコル

Daniel Wang
daniel@loopring.org

Jay Zhou
jay@loopring.org

Alex Wang
alex@loopring.org

Matthew Finestone
matt.finestone@gmail.com

https://loopring.org

2018 年 4 月 8 日

概要

ループリングは、分散型取引所を構築するためのオープンプロトコルである。ループリングは、取引と決済を執行するための一連のパブリック・スマートコントラクトとして機能する一方、注文のアグリゲーションと通信はオフ・チェーンで行われる。ループリングのプロトコルは自由度と拡張性が高く、取引機能を組み込んだ分散アプリケーション（dApp）の標準構成要素として機能する。その運用互換性基準はトラストレスかつ匿名の取引を容易にする。現存の分散型取引所のプロトコルと比べ、ループリングが大きく改善した点は、複数の注文による組み合わせを可能にし、2 種類のトークンからなる取引ペアの制限を取り除き、劇的な流動性の増加をもたらすことである。また、ループリングはユニークかつ堅固なフロントランニング防止ソリューションを採用している。フロントランニングとは、元のソリューションプロバイダに先立ってブロックにトランザクションを提出するといった不正行為である。ループリングはブロックチェーンに依存せず、スマートコントラクト機能を備えた如何なるブロックチェーンにも配置可能である。執筆時点では、すでにイーサリアム [1] [2] とクアンタム（Qtum）[3] のブロックチェーンで運用可能であり、NEO [4] の対応について現在構築中である。

1 導入

ブロックチェーンに基づくデジタル資産の急増に伴い、カウンターパーティー間の資産交換・取引の需要が急速に拡大している。特に近年において、数千種類の新しい仮想通貨のトークン（トークン化された伝統的な資産も含む）の出現によって、こうした需要がさらなる拡大を見せている。投機的動機のためであれ、手持ちのネイティブ・ユーティリティ・トークンを通してネットワークへのアクセス権を得るためであれ、仮想通貨間の交換・取引を実現する能力は、より大きいエコシステムの基礎となるであろう。確かに、これらの資産には潜在的なエネルギーがあり [5]、このエネルギーを解放するため（資本のロック解除）には、所有権の確保だけでなく（この点についてブロックチェーンによって永久に保障されている）、これらの資産を自由に移管・変換する能力もまた必要である。

このように、トラストレスなトークン（バリュー）交換・取引は、まさにブロックチェーン技術の切実な使用例である。しかしながら、現在のところ、仮想通貨の愛好者たちはほとんど伝統的な中央集権型取引所でトークンを取引している。ループリングが必要である理由は、「もし信用された第三者によって二重支払いが起きないように保証されなければならないならば」、ピア・ツー・ピアの電子キャッシュの「主な利点が失われてしまう」といった Bitcoin[5] の主張と同様に、信用された中央集権型取引所で取引されなければならないならば、分散型のデジタル資産の主な利点も失われてしまうからである。

哲学的な観点から見ても、中央集権型取引所で分散型のトークンを取引すること自体も理にかなっていない。分散型のプロジェクトが信奉する理念と一致してないからである。中央集権型取引所に他にも様々な現実的なリスクと限界があるが、これについて後述する。分散型取引所（DEXs）[6] [7] [8] はこれらの問題を解決してきており、ブロックチェーン技術の利用によるディスインターミディエーションを通じてセキュリティリスクの軽減に成功した例もすでに多く見られている。しかしながら、DEX の性能が新しい経済圏における重要なインフラになるにつれ、パフォーマンス面においてまだ改善の余地が大きい。ループリングは、独自の dApp 非依存型のオープン・プロトコルによって、前述したインフラのためにモジュラー・ツールを提供することを目的としている。

2 取引場の現状

2.1 中央集権型取引所の不足点

中央集権型取引所のリスクについて主に 3 つの点が挙げられる。1）安全性の欠如、2）透明性の欠如、3）流動性の不足。

安全性の欠如は、ユーザーが自分の秘密鍵（資金）の管理権を中央集権型取引所に委ねることに起因する。これによって、ユーザーは悪意のハッカーによる攻撃の犠牲になるリスクに晒されることとなる。このような中央集権型取引所の安全及びハッキングのリスクは周知されているにもかかわらず [9] [10] 、こ

れをしばしば、トークン取引を行う代わりに支払わなければならない対価として受け止められている。そこで現在、中央集権型取引所のサーバーはユーザーの資金を管理しているため、ハッカーにとって格好の攻撃対象となっている。また、取引所の開発者もユーザーの資金を管理するにあたり、誠実かつ予想外のミスを犯す可能性がある。要するに、ユーザーは中央集権型取引所に保管されている自分のトークンに対し管理権を有しないのである。

透明性が不足している状況の下で、ユーザーは不誠実な取引所による不正行為の危険に晒されることとなる。とりわけ取引所の運営者の悪意から、ユーザーが中央集権型取引所で自分の資産の代わりに借用証書（IOU）を取引しているといったこともあるうる。トークンがいったん取引所のウォーレットに送金されれば、取引所は当該トークンの管理権を有する代わりに IOU を発行する。これによってすべての取引は実際にはユーザー間の IOU's である、ということになる。出金する際に、ユーザーは取引所から IOU を自分のトークンに引き換え、引き出したトークンを自分の外部ウォーレットのアドレスに送金する、といったプロセスになる。このような送金・取引・出金のプロセスには透明性が乏しいだけでなく、取引所自体が閉鎖や破産したり、ユーザーのアカウントを凍結したりする恐れがある。また、ユーザーの資産を第三者に貸し出すなどの不正目的に利用する可能性もある。さらに、透明性の不足はユーザーの資産が全部流出するといった事態を招くことはないとしても、過剰の取引手数料、取引ピーク時の注文処理遅延、規制のリスクやフロントランニングなどリスクが依然として存在する

流動性の不足。取引所の運営者から見ると、流動性の分散は新しい取引所の参入を阻害する効果を持つ。これは二つの「勝者独り勝ちシナリオ」を根拠としている。第一に、ユーザーにとって全ての取引を一つの取引所で行うことが望ましいので、最も多い取引ペアを提供する取引所が勝ち残る。第二に、取引ペアの売買差額が小さい方は有利なので、注文が最も集中する取引所が勝ち残る。これによって新参者による初期の流動性確保が困難になり、競争を阻害することとなる。その結果、多くの取引所はユーザーからの苦情や重大なハッキング事件の多発にもかかわらず、依然として高い市場シェアを維持している。さらに、中央集権型取引所の市場シェアが拡大すればするほど、ハッキング攻撃の対象となる可能性も大きくなることも注意すべきである。

ユーザーの視点から見れば、流動性の分散化はユーザー・エクスペリエンスを著しく低下させる結果を招く。中央集権型取引所においては、ユーザーは取引所が提供する流動性プール、オーダーブック及び対応取引ペアに基づいて取引するしかない。トークンA をトークンB に交換するにあたり、ユーザーは両方のトークンを同時に対応している取引所か、あるいは別々の取引所で個人情報を開示して登録しなければならない。また、ユーザーはしばしば BTC や ETH などの主要通貨に基づいて初期または中間取引を行うことが必要となるが、この過程で生じた売買差額を負担しなければならない。最後に、取引が重大な遅延なしに執行されるための注文数が確保されていないこともありうる。取引所が大量取引の処理能力を誇示するとしても、その取引高や流動性は虚偽の情報でない保証はない [11]。

結果として、伝統的な金融システムと同様に流動性の断片化とエコシステムの分裂化をもたらし、取引は少数の中央集権型取引所に集中することとなる。ブロックチェーン技術がもたらすグローバル規模の流動性は、中央集権型取引所で価値を見出せない結果となるのである。

2.2 分散型取引所の不足点

分散型取引所と中央集権型取引所との違いの一つは、ユーザーは直接にブロックチェーンで取引することによって自分の秘密鍵（資産）の支配権を維持できることである。分散型取引所は仮想通貨自体のトラストレス技術を利用して、前述した多くの安全リスクを軽減することができる。しかし、パフォーマンスと構造的制限の問題がまだ残っている。

ユーザーは他の流動性プールと基準からカウンターパートを探さなければならないことから、流動性は問題となっている。ほとんどの分散型取引所または dApp が相互運用の際に統一した基準を採用しなければ、そして注文が広いネットワークで分散・伝播されなければ、流動性の断片化効果はまだ存在する。指値注文の流動性および弾力性（応じられた注文がどれほど早く再生されること）はユーザーの最善な取引戦略に重大な影響を及ぼしうる [12]。このような基準の不在は流動性の低下を招くだけでなく、安全上の問題を秘めている一連のスマートコントラクトに暴露されることとなる。

さらに、取引はブロックチェーンの上で行われる以上、分散型取引所には拡張可能性や執行（マイニング）の遅延、注文修正のコストなどのブロックチェーンの限界が内在することとなる。結果として、ブロックチェーンの上でコードを執行するとコスト（ガス）が生じるため、複数の注文がキャンセルされた場合に高額なコストが発生することから、ブロックチェーン・オーダーブックの拡張性が特に優れているわけではない。

最後に、ブロックチェーン・オーダーブックは公開されているため、発送された注文が次のブロックにマイニングされ、オーダーブックに記録されるのを待つ過程において、マイナーは注文の発送を見ることができる。この遅延によって、ユーザーはフロントランニングや、価格または約定が逆方向に動くといったリスクに晒されることとなる。

2.3 ハイブリッド・ソリューション

上記の理由から、完全にブロックチェーンに基づく分散型取引所は、その限界によって中央集権型取引所に対し競争上不利の立場に立たされた。こうして、オン・チェーンに内在するトラストレスの特性と、中央集権型の取引スピードや注文処理の柔軟性は、同時に享受できないのである。ループリングや 0x [13] のようなプロトコルはオン・チェーン決済とオフ・チェーン管理を組み合わせたソリューションを提供する。これらのソリューションはオープンなスマートコントラクトを中心に展開するが、同時にオフ・チェーンで複数の関数を実行し、ノードにネットワークのために決定的な役割を果たせるように柔軟性を持たせることによって、分散型取引所の拡張性の限界を克服する。しかしながら、ハイブリッドモデルにも欠点がある [14]。ループリング・プロトコルは、本ホワイトペーパーにおいてハイブリッド・ソリューションの欠点を補うために有意義な変更を提案する。

3 ループリング・プロトコル

ループリング自体は分散型取引所ではなく、複数のブロックチェーンの上で分散型取引所を構成するためのモジュラープロトコルである。我々は伝統的な取引所の構成要素を分解し、

代わりに一連のパブリック・スマートコントラクトと分散型の参加者を提供する。このネットワークには、ウォレット、中継器、流動性共有コンソーシアム・ブロックチェーン、オーダーブック・ブラウザー、リングマイナーおよび資産のトークン化サービスといった役割がある。それぞれを定義する前に、まずはループリング・オーダーについて理解する必要がある。

3.1 ループリング・オーダー

ループリング・オーダーは単指向注文モデル「UDOM」[15] と呼ばれるものによって表示される。UDOM は注文をトークン取引リクエストとして表示し、bids と asks の代わりに amountS/amountB(売却数量/買付数量)で表示する。個々の注文は 2 種類のトークン間の交換レートに過ぎないため、このプロトコルの強い特徴は、複数の注文を循環取引によってミキシング・マッチングすることができることである。単一の取引ペアの代わりに最大 16 個の注文を同時に執行することが可能であるため、流動性の劇的な増加をもたらすだけでなく、価格の改善も期待できる。

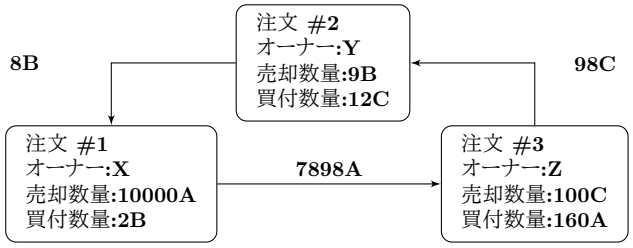

図 1: 3 つの注文からなるオーダー・リング

上記の図では、3 つの注文からなるオーダーリングの例を示している。それぞれの注文の売却トークン (tokenS) は、他の注文の買付トークン (tokenB) である。それぞれの注文は対となる反対注文が存在しなくても取引することが可能となり、こうして全ての注文を繋ぐループを作り出す。従来の取引ペアによる取引は実質的にオーダーリングの特例となるが、もちろん執行可能である。

定義 **3.1** (オーダーリング) $C_0, C_1, \cdots, C_{n-1}$ は n 個の異なる種類のトークンとし、$O_{0\to1}, \cdots, O_{i\to i\oplus1}, \cdots, O_{n-1\to0}$ は n 個の注文とする。これらの注文は一つのオーダーリングを形成する:

$$O_{0\to1} \to \cdots \to O_{i\to i\oplus1} \to \cdots \to O_{n-1\to0},$$

n はオーダーリングの長さである場合、$i \oplus 1 \equiv i+1 \mod n$ となる。

全ての注文において、利用者が指値した初期レート以上のレートで執行されることが可能の場合、オーダーリングは有効となる。オーダーリングの有効性を検証するために、ループリングプロトコル・スマートコントラクトはリングマイナーからオーダーリングを受信するが、同オーダーリングにあるすべての注文の初期交換レートの積が 1 以上とならなければならない。

アリスとボブはトークンA とB を取引しようとすると仮定する。アリスは 15 個のトークンA を持っているが、それを 4 個のトークンB に交換したいと思っている。ボブは 10 個のトークンB を持っているが、それを 30 個のトークンA に交換したいと思っている。

この場合、買い手と売り手はそれぞれ誰となるだろうか?それは、どれのトークンを参照基準にするかによる。トークンA を参照基準とする場合、アリスは $\frac{15}{4} = 3.75$A の価格でトークンB を買い付けることになり、ボブは $\frac{30}{10} = 3.00$A の価格でトークンB を売却することになる。トークンB を参照基準とする場合、アリスは $\frac{4}{15} = 0.26666667$B の価格でトークンA を売却することになり、ボブは $\frac{10}{30} = 0.33333334$B の価格でトークンA を買い付けることになる。従って、買い手と売り手が誰になるかは任意である。

前者の場合、アリスの買付指値 (3.75A) はボブの売却指値 (3.00A) より高いが、後者の場合、ボブの買付指値 (0.33333334B) はアリスの売却指値 (0.26666667B) より高い。明らかに、買い手の指値が売り方の指値と同等または上回る場合に約定が可能となる。

$$\frac{\frac{15}{4}}{\frac{30}{10}} = \frac{\frac{10}{30}}{\frac{4}{15}} = \frac{15}{4} \cdot \frac{10}{30} = 1.25 > 1 \tag{1}$$

よって、ループを構成する n 個の注文が全部または一部約定するためには、全ての注文の買付交換レートの積が 1 より大きくなっているかどうかを知る必要がある。1 より大きい場合、n 個の注文はすべて一部または全部約定することとなる [16]。

さらに 3 人目のカウンターパーティであるチャーリーが加わるとする。チャーリーの加入によって、アリスは x_1 個のトークンA を y_1 個のトークンB に、ボブは x_2 個のトークンB を y_2 個のトークン C に、チャーリーは x_3 個のトークンC を y_3 個のトークンA に交換したいといった要求が満されると仮定する。そして必要とされるトークンもすべて揃っているとする。この場合、下記の条件を満せば、取引が可能となる:

$$\frac{x_1 \cdot x_2 \cdot x_3}{y_1 \cdot y_2 \cdot y_3} \geq 1 \tag{2}$$

ループリング・オーダーの詳細について 7.1 で後述する。

4 エコシステムの参加者

下記のエコシステムの参加者は共同で分散型取引所に求められる機能を提供する。

- ウォレット:一般的なウォレット・サービスまたはインターフェースであり、ユーザーがトークンにアクセスし、また注文をループリングのネットワークに送信するための方法を提供する。ウォレットはリングマイナーと手数料を分担することによって、注文を生成するためのインセンティブが得られる(詳細は 8を参照)。我々は、将来にはトークン取引は安全性が確保されたユーザー個人のウォレットで行われるようになることを確信しているため、流動性プールをループリング・プロトコルで結びつけることが最重要である。

- 流動性共有コンソーシアム・ブロックチェーン/リレーメッシュ:注文と流動性の共有のためのリレーメッシュ・ネットワークである。ノードがループリングリレー・ソフトウェアを実行する際に、既存のネットワークに参加し、コンソーシアム・ブロックチェーンを介して他の中継器(リレー)と流動性を共有することができる。我々が構築しているコンソー

シアム・ブロックチェーンは世界初の実装であり、ほぼリアルタイム(1-2 秒ブロック時間)で注文を共有することが可能であるだけでなく、新しいノードがより迅速にダウンロードできるように古い履歴を調節することができる。特に、中継器はこのコンソーシアムに参加する必要がなく、他の中継器と流動性を共有せずに単独で機能することができ、または独自の流動性共有ネットワークを構築・運営することも可能である。

- 中継器/リングマイナー: 中継器は、ウォレットまたはリレーメッシュから注文を受信し、公開されたオーダーブックと取引履歴を管理し、そして(任意のオフチェーン媒体を介して)他の中継器および/またはリレーメッシュ・ノードに注文をブロードキャストすることができるノードである。リングマイニングは中継器の特性であり、要請ではない。リングマイニングの計算量が大きく、完全にオフチェーンで行われる。リングマイニング機能を備えた中継器を「リングマイナー」("Ring-Miners") と呼ぶが、リングマイナーは異種の注文をつなぎ合わせてオーダーリングを生成する。中継器は、(1) 相互間の通信方法、(2) オーダーブックの作成方式、および (3) オーダーリングのマイニング方法(マイニングアルゴリズム)を自由に決めることができる。

- ループリングプロトコル・スマートコントラクト **(LPSC)**: LPSC は、リングマイナーから受信したオーダーリングをチェックし、ユーザーの代わりにトークンをトラストレスで決済・移動し、手数料をもってリングマイナーとウォレットにインセンティブを与え、イベントを発行する一連の公開・自由のスマートコントラクトである。中継器/オーダーブック・ブラウザは、これらのイベントを待ち受け、オーダーブックと取引履歴を最新の状態に維持する。詳細について付録 Aを参照。

- 資産トークン化サービス **(ATS)**: ループリングにおいて、直接的に交換できない資産の間の架け橋である。ATS は、信頼できる企業や組織によって運営されている中央集権型のサービスである。ユーザーは資産(現物や法定通貨、または他のブロックチェーン上のトークン)を預託した代わりにトークンを発行されるが、同トークンをもって預託資産と引き換えることができる。ループリングプロトコルは(適切なソリューションが開発されるまで)クロスチェーン取引のプロトコルではないが、ATS は ERC20 のトークンと物理的な資産および他のブロックチェーン上の資産 [17] の間の取引を可能にする。

5 取引プロセス

1. プロトコルの承認: 図 2 では、トークンを取引したいユーザーY は、売却数量のトークンB(amountS)の売却について LPSC に承認を与える。ユーザーのトークンは承認によってロックされるわけではなく、ユーザーは注文の処理中においてトークンを自由に移動することができる。

2. 注文の生成: 最新のトークンB とトークンC 間の交換レートとオーダーブック(板情報)は、中継器またはネットワークに接続された他のエージェント(オーダーブックブラウザなど)によって提供される。ユーザーY は、統合されたウォレット・インターフェースを介して、売却数量 (amountS) と買付数量 (amountB) 等のパラメータを指定し、指値注文を提出する。一定数の LRx をリングマイナーに支払う手数料として注文に追加することができる。支払う手数料が高いほど、注文はリングマイナーによって処理される可能性が高くなることを意味する。注文のハッシュ値はユーザーY の秘密鍵で署名される。

3. 注文のブロードキャスト: ウォレットは注文とその署名を 1 つまたは複数の中継器に送信する。中継器はその公開されたオーダーブックを更新する。このプロトコルは、先着順などといった特定の方法によるオーダーブックの作成を要求しない。その代りに、中継器は独自のオーダーブックを作成する際に、設計に関する決定権を有する。

4. 流動性の共有: 中継器は任意の通信媒体を介して他の中継器に注文情報をブロードキャストする。繰り返しになるが、ノード間の交互方法には自由度がある。ネットワークの接続性を一定レベルに保つため、コンソーシアム・ブロックチェーンを使用した組み込み式の流動性共有リレーメッシュが用意されている。前章で述べたように、このリレーメッシュはネットワークの通信速度と協調性を保てるように最適化されている。

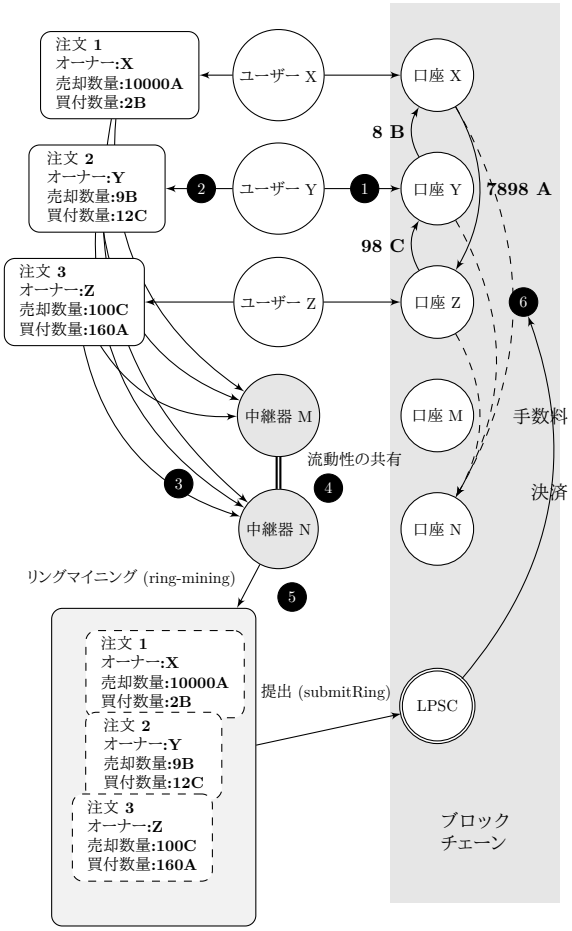

図 2: ループリング取引プロセス

5. リングマイニング(注文マッチング): リングマイナーは、指定された交換レート以上のレートで注文を他の複数の注

文と全部または部分的にマッチングさせることを試みる。このリングマイナーの働きこそが、このプロトコルはどのペアよりも高い流動性を提供できる主な理由である。執行された交換レートはユーザー Y が指定したものより高い場合、マージンはオーダーリング内のすべての注文の間で共有される。リングマイナーは報酬として、マージンの一部を請求する（Margin-Split、LRx はユーザーに返還される）かまたは単に手数料として LRx を受け取るかを選択することになる。

6. 検証と決済：オーダーリングは LPSC によって受信される。LPSC はリングマイナーから提供されたデータを検証し、（オーダーリング内の約定成功率およびユーザーウォレット内のトークン残高によって）オーダーリングが全部または部分的に応じられるか否かを確認するために数回にわたってチェックを行う。全てのチェックが問題なく完了した場合、LPSC は自動的にトークンをユーザーに転送すると同時に、リングマイナーとウォレットに手数料を支給する。ユーザーY の残高が不足していると LPSC が判定した場合、注文の数量は縮小される（スケールダウン）ことになる。スケールダウンは注文の取消と異なる。スケールダウンは十分な資金が特定のアドレスに預託されれば自動的に元の数量にスケールアップされるが、注文の取消は一方的なマニュアル操作であり、復元することができない。

6 運用上の柔軟性

ループリングのオープンスタンダードは参加者による運用にあたって大きな柔軟性を与えていることは、注目に値すべきである。参加者は自由に新しいビジネスモデルを構築し、ユーザーに価値を提供することができ、同時に（自らそう選択した場合に）取引高などの指標で LRx の手数料を得る。エコシステムはモジュール式であり、より多くのアプリケーションの参加をサポートするために作られた。

6.1 オーダーブック

中継器は、ユーザーの注文の表示とマッチングの方法に関し、様々なオーダーブックを設計することができる。我々が実装した最初のオーダーブックは、指値注文が価格のみに基づいて配置されるといった OTC モデルに従ったものである。言い換えれば、注文のタイムスタンプはオーダーブックと関係がない。しかしながら、中継器は、典型的な中央集権型取引所の注文マッチング・エンジンをエミュレートし、各注文はタイムスタンプを尊重すると同時に価格に基づいてランク付けするというようなオーダーブックを設計することも可能である。中継器にこのタイプのオーダーブックを提供する傾向がある場合は、同中継器はウォレットを所有または統合し、ウォレット・オーダーを同中継器のみに送信するようにすることができ、これによって同中継器は時間に基づいて注文をマッチングすることが可能となる。

一方、他の分散型取引所のプロトコルは時々、中継器に一定のリソース（発注するための初期トークン残高）があることを要請するに対し、ループリングの中継器は、トークンの初期残高無しに、マッチする注文を探し当てるだけで取引を成立させることが可能である。

6.2 流動性の共有

中継器は、どのように流動性（注文）を共有するのかを自由に設計できる。ループリングのコンソーシアムブロックチェーンはこれを達成するための一つの解決策であり、中継器はエコシステムの中で自由に相互通信することができる。コンソーシアムブロックチェーンに参加することを除き、中継器は自由にネットワークを構築・運営し、独自のルール/インセンティブを定めることができる。また、タイムスタンプ依存型のウォレット実装でも説明したように、中継器は単独で機能することもできる。もちろん、ネットワーク効果の追求において他の中継器と通信することには明らかなメリットがあるが、ビジネスモデルの多様化は、様々な方法で独自の共有方法の設定と手数料の分担を可能にするといったメリットがある。

7 プロトコル仕様

7.1 「注文」の詳細

注文とは、ユーザーが意図する取引を表すデータの集まりである。ループリングにおける注文は、UDOM(the Uni-Directional Order Model、単指向性注文モデル) に従うように定義されている。詳細を以下に示す。

```
message Order {
  address protocol;
  address owner;
  address tokenS;
  address tokenB;
  uint256 amountS;
  uint256 amountB;
  unit256 lrcFee
  unit256 validSince; // システム時刻
  unit256 validUntil; // システム時刻
  uint8   marginSplitPercentage;  // [1-100]
  bool    buyNoMoreThanAmountB;
  uint256 walletId;
  // Dual-Authoring address
  // 二重承認アドレス
  address authAddr;
 // v, r, sで署名を構成する
  uint8   v;
  bytes32 r;
  bytes32 s;
  // 二重承認用秘密鍵、
  // 注文のハッシュ値計算用ではない、
  // よって署名されていない
  string  authKey;
  uint256 nonce;
}
```

注文の発信元を確実にするため、authAddr を除くパラメータのハッシュ値に対してユーザーの秘密鍵で署名する。authAddr パラメータは、ある注文があるオーダーリングの一部であることを証明し、署名するために使用されるものである。また、それにはフロントランニングを防止する役割もある。詳細は 9.1を参照。署名は、v, r, s の 3 つの変数で表され、ネットワー

クを介して注文パラメータとともに送信される。これにより、注文データは永久に不変であることを証明する。注文データについて、変更されることは有り得ないが、プロトコルはアドレスの残高や他の変数に基づいて最新の状態を計算できる。

UDOM には (本来であれば浮動小数点数でなければならない) 指値価格の情報は含まれていない。その代りに、amountS/amountB(売却数量/買付け数量)という形式で注文価格を表す。これをレート (rate) と呼ぶことにし、略して r とも示す。レートは浮動小数点数ではなく、符号なし整数で示される。これは、すべての計算の中間結果を符号なし整数として保持し、計算精度を向上させるためである。

7.1.1 買付数量

リングマイナーが注文をリング・マッチングするときに、より良いレートで注文が執行される可能性がある。つまり、ユーザーは、注文時に指定した買付数量(amountB)よりも多くのtokenB を受け取るする可能性がある。しかし、もしbuyNoMoreThanAmountB のパラメーラ値がTrue として設定されている場合、ループリングプロトコルは、ユーザーが注文で指定した買付数量(amountB)以上のtokenB を受け取らないように動作する。したがって、UDOM 中の buyNoMoreThantokenB パラメータは、注文が完全に約定されたとみなされるタイミングを決定する。buyNoMoreThantokenB というパラメータは、売却数量(amountS)または買付数量(amountB)のいずれかに上限を設定するものであり、従来の注文よりも細かい取引意思を表現することができる。

例えば、amountS = 10 かつ amountB = 2 の場合、レート r = 10/2 = 5 となる。これはつまり、ユーザーは、1 つの tokenB につき 5 つのtokenS で交換したい意志を持っていることを示す。もしも、リングマイナーがこの注文に対して、レートが 4 の反対注文を探り当て、マッチングが成功した場合、ユーザーは、2 つではなく、2.5 の tokenB を受け取ることになる。しかし、ユーザーは、2 つのtokenB だけ受け取りたく、buyNoMoreThanAmountB のパラメーター値をTrue に設定した場合、LPSC は、1 つの tokenB を入手するのに 4 つのtokenS を売り出すトランザクションを実行する。その結果、ユーザーにとっては、2 つのtokenS を節約できたことになる。注意していただきたいのは、説明をシンプルにするために、上記ではマイニング手数料 (詳細は 8.1 を参照) を無視している。

注文の内容を下記の簡易な形式で表現することもできる。

```
Order(amountS,tokenS,
      amountB,tokenB,
      buyNoMoreThantokenB)
```

例えば、従来の取引場における ETH/USD のペア市場では、下記の 1 と 3 を表現できるが、残りの 2 と 4 を表現できない。

1. 10 ETH を 300USD/ETH のレートで売却することは:`Order(10, ETH, 3000, USD, False)` で表現する。
2. ETH を 300USD/ETH のレートで売却し 3000USD を入手することは:`Order(10, ETH, 3000, USD, True)` で表現する。
3. 10 ETH を 300USD/ETH のレートで買付することは:`Order(3000, USD, 10, ETH, True)` で表現する。
4. 3000USD で 10ETH 以上でなるべく多くの ETH を購入することは:`Order(3000, USD, 10, ETH, False)` で表現する。

7.2 リング検証

ループリングのスマートコントラクトは交換レートや売買数量の計算を行わないが、リングマイナーからこれらの値に対する検証結果を受信し、確認する必要がある。これらの計算はリングマイナーによって行われる主な理由は下記の 2 つである。(1) イーサリアムにおける solidity[18] のようなスマートコントラクト用のプログラミング言語は小数点計算に対応していない。特に $pow(x, 1/n)$ (小数の n 次累乗のルートを求める計算)。(2) ブロックチェーンの計算コストを削減するために、計算処理はオフチェーンで行うほうが望ましい。

7.2.1 サブリング・チェック

このステップは、オーダーリングの中で新しい注文を追加、執行することによって、全てのマージンを独り占めするような不公平なアービトラージ取引を防ぐ。本質的に、有効なオーダーリングがリングマイナーによって見つけられると、オーダーリングに新しい注文を追加すれば、ユーザーにとってのマージン(レート割引)を完全に吸収してしまうことができる。下記の図 3に示すように、巧みに計算された $x1$, $y1$, $x2$, $y2$ は、すべての注文レートの積を正確に 1 にするので、ユーザーにとってのレート割引は完全になくなる。

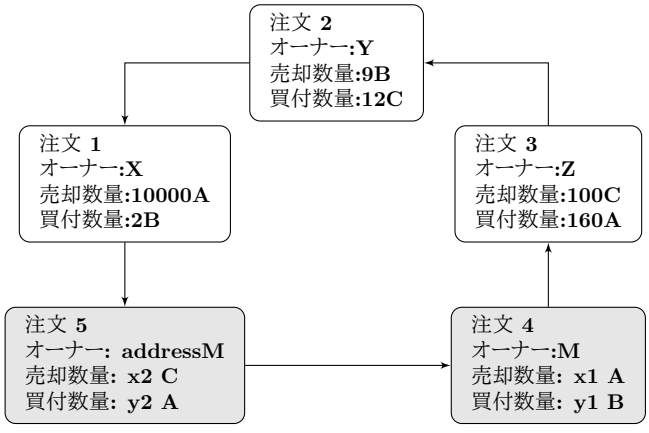

図 3: サブリングを含むオーダーリングの例

ネットワークにゼロバリューを追加することはゼロリスクであるが、これはリングマナーによる不正行為としてみなされる。これを防ぐために、ループリングにおいては、有効なオーダーリングにサブリングを含めることは許されていない。これを確認するために、LPSC は同じトークンが買付注文または売却注文で 2 回登場しているか否かのチェックを行っている。例えば、上の図では、トークンA が 2 回登場しており、チェックの結果としてこれは不正行為であり、許されないことになる。

7.2.2 約定率の計算

オーダーリングにおけるレートの計算は、上述の理由により、リングマイナーによって行われる。マイナーによって計算されたレートの正しさをチェックするのは LPSC である。まず、リングマイナーが各注文に対して執行できる買付レートが、ユーザーが設定した元の買付レート以下であることを確認する。これにより、少なくともユーザーが注文時に設定したレートと等しい、或いはより得するレートで、取引が執行されることを保証する。取引レートが確定されると、LPSC はオーダーリング内の各注文が同じ割引されたレートを共有するようにする。たとえば、割引されたレートが γ である場合、各注文の価格は次のようになる:

$r_{0\to1}\cdot(1-\gamma)$, $r_{1\to2}\cdot(1-\gamma)$, $r_{2\to0}\cdot(1-\gamma)$, かつ、以下を満たす:

$$r_{0\to1}\cdot(1-\gamma)\cdot r_{1\to2}\cdot(1-\gamma)\cdot r_{2\to0}\cdot(1-\gamma)=1 \tag{3}$$

したがって:

$$\gamma = 1-\frac{1}{\sqrt[3]{r_{0\to1}\cdot r_{1\to2}\cdot r_{2\to0}}}. \tag{4}$$

もしトランザクションに n 個の注文を含んでいる場合、割引されたレートは:

$$\gamma = 1-\frac{1}{\sqrt[n]{\prod_{i=0}^{n-1} r^i}}, \tag{5}$$

ここでは、r^i は i 個目の注文の回転率である。明らかに、ディスカウントレート (discount rate) が $\gamma \geq 0$ のときのみ、これらの注文が約定されることができる。そして、i 個目の注文 (O^i) の実際のレートは:$\hat{r^i} = r^i\cdot(1-\gamma)$, $\hat{r^i} \leq r^i$ となる。

前述の例を振り返てみると、アリスは 15 つのトークンA を所有しており、これらを 4 つのトークンB に交換したいと思っている。ボブは 10 のトークンB を所有しており、これらを 30 のトークン A と交換したいと思っている。トークンA を基準に考えると、アリスは $\frac{15}{4}=3.75$A の価格でトークンB を買付しようとしている。一方で、ボブは $\frac{30}{10}=3.00$A の価格でトークンB を売却しようとしている。割引(ディスカウント)は:$\frac{150}{120}=1.25$ よって、$\frac{1}{1.25}=0.8=(1--\gamma)^2$。したがって、この取引において、両者にとって公平なレートは:$\sqrt{0.8}\cdot 3.75\approx 3.3541$ トークンA/トークンB である。

ボブは 4 つのトークン B を渡し、13.4164 のトークンA を受け取るため、ボブが予期していた 12 よりも多くトークンA を受け取る。一方で、アリスは 4 つのB を受け取るのに 13.4164 のトークンA のみを渡す。これはアリスが予期していた 15 よりも少なくトークンA を渡すことになる。

注意していただきたいのは、一部のマージンはインセンティブとしてマイナー(ウォレット)に渡すことになる。(詳細は 8.1 を参照)。

7.2.3 約定状況のトラッキングと注文の取消

ユーザーは、既に提出した注文の詳細や、キャンセルしたい数量等の情報を含んだ特殊なトランザクションを LPSC に送信することによって、既に提出した注文の全部または一部を取消することができる。LPSC はこれを考慮に入れ、キャンセルする数量を保管し、`OrderCancelled` というイベントをネットワークに送信する。LPSC は、注文のハッシュ値を注文の ID として使用することで、該当する注文を特定し、執行された数量と取消された数量をトラッキングし続ける。これらのデータは公開されており、また、`OrderCancelled` / `OrderFilled` のイベントはこれらのデータが変更するたびにネットワークに送信される。オーダーリング決済のステップでは、これらの値をトラッキングすることは、LPSC にとってクリティカルなことである。

LPSC はまた、`OrdersCancelled` というイベントで、特定の取引ペアの注文を取消することができ、`AllOrdersCancelled` というイベントで、特定のアドレス中の全ての注文を取消することができる。

7.2.4 注文の縮小(スケーリング)

注文は、執行された数量、取消された数量、注文送信者の口座中の現在残高によって、スケーリングされる。この特性に従い、このプロセスはすべての注文のなかで最小約定数量をもつ注文を探し出し、それを参照として、オーダーリング中のすべての注文のスケーリングを行う。

最小約定数量をもつ注文を探し出せば、全ての注文の約定数量の計算が容易になる。例えば、i 個目の注文が最小約定数量をもつ注文だとすれば、それぞれの注文で約定される売却数量 $\hat{s}$ と、それぞれの注文で約定される買付数量 $\hat{b}$ は、以下のように計算できる:

$$\begin{aligned}
&\hat{s}^i = \overline{s}_i,\ \hat{b}^i = \hat{s}^i/\hat{r}^i,\ ;\\
&\hat{s}^{i\oplus1} = \hat{b}^i,\ \hat{b}^{i\oplus1} = \hat{s}^{i\oplus1}/\hat{r}^{i\oplus1};\\
&\hat{s}^{i\oplus2} = \hat{b}^{i\oplus1},\ \hat{b}^{i\oplus2} = \hat{s}^{i\oplus2}/\hat{r}^{i\oplus2};\\
&\ldots
\end{aligned}$$

そのなかで、$\overline{s}_i$ とは約定された後の口座残高を指す。

実際の運用のなかで、オーダーリング中の任意の注文が最小約定数量を持つ注文だと仮定し、オーダーリングのなかでループ処理を行って、最大 2 回の計算を行うだけでそれぞれの注文の約定数量を計算できる。

例えば、最小約定数量が注文の中で指値した数量の 5% である場合、オーダーリング中のすべての注文の約定数量は 5% に縮小(スケールダウン)される。約定後、最小約定数量をもつ注文はすでに全部約定されるはずである。

7.3 オーダーリング決済

オーダーリングが前述のすべてのチェックを通過していれば、そのオーダーリングはクローズドとなり、決済に移ることができる。これは、n 個の注文はひとつの閉じられたループで繋げられることを意味する。その様子を下記の図に示す。

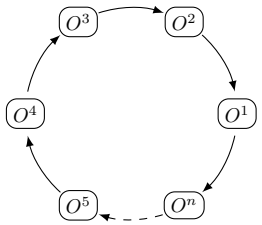

図 4: オーダーリング決済

決済を完了させるのに、TokenTransferDelegate(トークン移動/送金用のデリゲート)という名のスマートコントラクトがLPSC によって使用される。このようなデリゲートの導入はスマートコントラクトのアップグレードをより容易にする。なぜなら、注文は、プロトコルのバージョンを考慮する必要がなく、このデリゲートをオーソライズさえすれば良いのである。

オーダーリングのそれぞれの注文について、約定状況により、一定のtokenS を前後の注文に支払うことになる。また、マイナーにも選択された手数料モデルに従い手数料をを支払う。最後に、全ての決済が完了すれば、RingMined(オーダーリングマイニング完了)といういイベントを生成する。

7.3.1 イベントの生成

ループリングプロトコルはイベントを生成することにより、オーダーブックの最新状況を、中継器、オーダーブラウザー、及びその他のエコシステム参加者にいち早く知らせる。イベントの種類を以下に示す。

- **OrderCancelled**:特定の注文が取消された際のイベント。
- **OrdersCancelled**:特定のユーザーのアドレス中、一つの取引ペアに関する注文が取消された際のイベント。
- **AllOrdersCancelled**:特定のユーザーのアドレス中、すべての取引ペアに関する注文が取消された際のイベント。
- **RingMined**:特定のオーダーリングの決済が完了した際のイベント。このイベントには、リング中におけるトークン移動履歴データが含まれている。

8 LRx トークン

「LRx」は、ループリングトークンの一般表記である。「LRC」はイーサリアムのブロックチェーン上の、「LRQ」はクアンタム(Qtum) 上の、「LRN」は NEO 上のトークンである。今後は、LRx が他のパブリックブロックチェーンでトークンが開発されるにつれ、順次に紹介する予定である。

8.1 手数料モデル

ユーザーは注文を作成するとき、リングマイナーに支払う手数料として、LRx の数量とリングマイナーが請求できるマージン分担比率 (marginSplitPercentage) を指定する。これをマージンスプリットと言う。リングマイナーは LRx とマージンスプリットのどちらを報酬として受け取るかを選択することができる。

以下の図は典型的なマージンスプリットの例を示している:

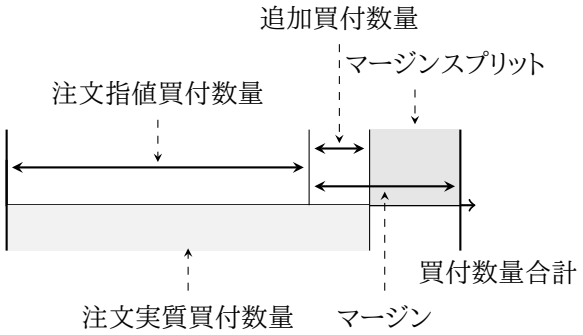

図 5: マージンスプリットが 60%の場合

オーダーリング状のマージンが少なすぎる場合、リングマイナーは LRx を報酬として選択し、反対にマージンが十分に大きく、マージンスプリットが LRx を大きく上回る場合、リングマイナーはマージンスプリットを選択する。但し、リングマイナーがマージンスプリットを選択した場合、ユーザー(発注者)に対して、ユーザーが手数料としてリングマイナーに支払うはずだった LRx の数量と同等な LRx を支払う必要がある。これのことはリングマイナーがマージンスプリットを選択する場合の敷居を LRx を選択した場合の 2 倍まで高めることとなり、結果として LRx を報酬として選択する傾向を強化することとなる。よって、リングマイナーはマージンの高いオーダーリングでより少ない収入を得ることのトレードオフとして、マージンの低いオーダーリングで安定的に収入を得ることが可能となる。ループリングの手数料モデルは、マージンの高いオーダーリングが市場の成長・成熟につれ減少することによって、固定の LRx 手数料は必然的にインセンティブとして必要されるようになるといった期待に基づいている。

下記の図に示されている状況が予想される。

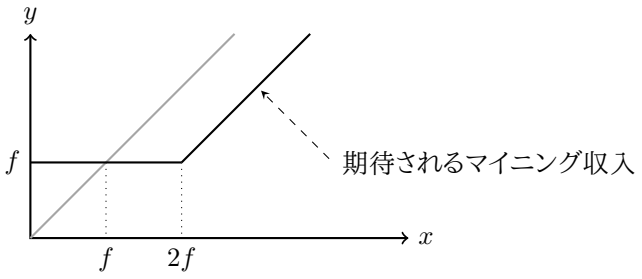

図 6: ループリングの手数料モデル

f を LRx 手数料、x 軸をマージンスプリット、y 軸をマイニングによる収入とする。$y = max(f, x - f)$ は黒実線で示したとおりであるが、もし LRx 手数料 =0 であれば、$y = max(0, x - 0)$ は灰実線で示している。

結果は以下の通りとなる:

1. マージンスプリットが 0 の場合、リングマイナーはまだインセンティブ効果がある LRx 手数料を選択する。
2. LRx 手数料が 0 の場合、結果は灰実線となり、収入は一般線形モデルに基づくこととなる。
3. マージンスプリット収入が 2x(LRx 手数料)を超える場合、リングマイナーはマージンスプリットを選択して、ユーザーに LRx を支払う。

注意すべきなのは、LRx 料金が 0 でない場合、リングマイナーがどのオプションを選択しても、リングマイナーと発注者の間に LRx の移動が必ず発生する。すなわち、リングマイナーは LRx 手数料を獲得するか、または LRx 手数料を発注者に支払い、マージンスプリットを受け取ることとなる。

それと、リングマイナーはウォレットと一定比率の手数料を共有する。ユーザーがウォレットを通して注文を送信し、それが執行されると、ウォレットは一定の LRx 手数料またマージンスプリットを報酬として受け取る。こうした設計はモジュール式であり、独自のビジネスモデルまたは実装は可能であるが、我々の慣例はウォレットが手数料の約 20%-25% を受け取ることである。ウォレットはユーザー基盤があるためループリングプロトコル・インテグレーションの主要ターゲットではあるが、収入源がほとんどないからである。

8.2 分散型ガバナンス

メンバー間の協調による効率的な目標達成という意味では、ループリングプロトコルは根本的にソーシャルプロトコルである。この点は、一般の暗号通貨界隈のプロトコルと異なるわけではなく、実際にその有用性は調整問題 [19]、グリムトリガー均衡、限定合理性等と同様なメカニズムによって保護されている。つまり、LRx トークンは手数料として支払うためのものだけでなく、多数のネットワーク参加者の報酬金を調整するためにも使用される。このような調整機能はどのプロトコルが幅広く採用されるために必要であるが、取引所は分散型エコシステムにおける流動性の向上によって成否が決定されるため、こうした調整機能は取引所のプロトコルにとって特に重要である。

LRx トークンは、分散型ガバナンスを通じてプロトコルの更新のためにも使用される。スマートコントラクトの更新は継続性と安全性を確保し、非互換性による流動性低下のリスクを軽減するためにトークン所有者によって管理される。スマートコントラクトはいったんデプロイされると変更できなくなるため、dApp またはエンドユーザーが古いバージョンのものと交互し、更新されたコントラクトから除外されるといったリスクが存在する。プロトコルは市場の需要と基盤となるブロックチェーンに適応しなければならないため、アップグレード可能性はプロトコルの成功にとって非常に重要である。LRx のステークホルダーによる分散型ガバナンスの下で、スマートコントラクトは dApp やエンドユーザーを妨害せず、またはスマートコントラクトの抽象化に過度に依頼せずに更新することが可能となる。LRx トークンの発行量は固定である。例えば LRC の場合、一部はループリング財団に帰属し、また一部はコミュニティーに提供するための基金に配分されることとなる。

しかしながら、LRx トークンの所有者はプロトコルの発展方向を決める唯一のステークホルダーではない。ほかには、中継器・リングマイナーや開発者などもエコシステムの不可欠の構成員部であり、彼らの意見も無視してはならない。実際に、これらの構成員はそれぞれの役割を果たすには LRx の所有が必要とされないことを鑑みると（伝統的な買い手・売り手やマーケットメーカーが存在しないため、初期におけるトークンの保有は強制的ではない）、彼らの利益を確保するために代替的な方法を許可しなければならない。さらに、「単なる」トークンによる投票は、オンチェーン・オフチェーンにかかわらず、低い投票率やトークン所有権の集中はリスクをもたらすため、意見の相違を和らげる方法としては不十分である。従って、我々の目標は、多層からなるガバナンスモデルおよび共有知識に基づく意思決定のプロセスのルールを構築することである。これを達成するために、あらゆる参加者や（可能ならば）予定されたプロトコルのフォーカスポイントからの意見を収集する協調機構から、一定のサポートを得ることが期待できる。この目標が実現すれば、ループリング財団は必然的にプロトコルの開発者役から執事役へと進化するであろう。

9 詐欺と攻撃に対する防衛措置

9.1 フロントランニングの防止

分散型取引所におけるフロントランニングとは、別のノードの取引ソリューションをコピーし、保留中のトランザクションプール（メモリプール）にあるコピー元のトランザクションに先だってマイニングすることである。これはより高い取引手数料（ガス価格）を指定することで可能となる。ループリング（または他の注文マッチングプロトコル）において、主なフロントランニング・スキームは「注文の横取り (order-filch)」であり、すなわち、フロントランナーが保留中のオーダーリング決済トランザクションから 1 つ以上の注文を横取りすることである。また、ループリングにおいて保留中のトランザクションからオーダーリングを丸ごと横取りすることさえありうる。

サブミットリング（submitRing）トランザクションがまだ確認されておらず、未処理のトランザクションプールに残っている場合、誰もが簡単にこのトランザクションを見分け、`minerAddress` を横取り屋である自分のアドレス（`filcherAddress`）に置き換えることができる。次に、`filcherAddress` でペイロードを辞退して、オーダーリングの署名を置き換えることが可能になる。最後に、横取り屋は、より高いガス価格を設定し、新しいトランザクションを提出すると、ブロックマイナーが元の submitRing トランザクションの代わりに自分のトランザクションを次のブロックに書き込むのを待てば良いのである。

この問題に対する従来の解決策には重要な欠点がある：より多くの取引が必要となり、その結果より多くのガス費用がかかることと、オーダーリングを決済するのに、少なくとも 2 倍のブロックの数が必要となることである。我々が提供する新しいソリューションである「二重承認 (Dual Authoring)」[20] には、注文の決済とリングマイニングについてそれぞれ承認を必要とするといった二段階による承認を設定する仕組みを含む。

二重承認のプロセスは下記の通りである。

1. ウォレット・ソフトウェアは各注文についてランダムな公開鍵/秘密鍵のペアを生成し、その鍵ペアを注文の JSON スニペットに入れる。（別の方法として、公開鍵の代わりに公開鍵から派生したアドレスを使用してバイト・サイズを減らす方法がある。このようなアドレスを「`authKey`」と表記し、認証アドレスと一致する秘密鍵を「`authAddr`」と表記する。）

2. `r`, `v`, `s`, 及び `authKey` を除いて、注文にあるすべての項目に基づいてハッシュ値を計算し、そのハッシュ値に対してオーナーの秘密鍵（`authKey`）で署名する。

3. ウォレットはリングマイニングのために`authKey` とともに注文を中継器に送信する。リングマイナーは、`authKey`

とauthAddr が正しくペアになり、注文の署名がオーナーのアドレスに関して有効であることを検証する。

4. オーダーリングが識別されると、リングマイナーは各注文のauthKey を使用して、リングのハッシュ値、minerAddress およびすべてのマイニングパラメータに署名する。オーダーリングに n 個の注文が含まれている場合、n 個のauthKeys によって n 個の署名が作成される。これらの署名を「authSignatures」と言う。また、リングマイナーはminerAddress の秘密鍵を使用して、すべてのマイニングパラメータとともにリングのハッシュに署名する必要がある。

5. リングマイナーは、すべてのパラメータと余分のauthSignatures で submitRing 関数を呼び出す。authSignatures はオンチェーンのトランザクションの一部ではないため、リングマイナー以外の当事者に知られていないことを注意しよう。

6. ループリングプロトコルは各注文に対応するauthAddr に対し、それぞれのauthSignature を検証するが、authSignature が存在しない或いは無効の場合は拒否される。

結果として:

- 各注文の(オーナーアドレスの秘密鍵による)署名は、authAddr を含む注文が変更不能であることを保証する。
- リングマイナーの(minerAddress の秘密鍵による)は、提供された場合において、誰もそのリングマイナーの ID でオーダーリングをマイニングすることができないことを保証する。
- authSignatures は、minerAddress を含むオーダーリング全体が変更不能であることと、注文の横取りが不可能であることを保証する。

二重承認は、オーダーリングの決済が 1 つのトランザクションの中で決済されることを確保すると同時に、「オーダーリング横取り」と「注文横取り」を防止する。それに加えて、二重承認は中継器が 2 つの方法によって注文を共有することを可能にしている。ノンマッチャブル共有 (nonmatchable sharing) とマッチャブル共有 (matchable sharing)。デフォルトは、ループリングは OTC をモデルに運営し、指値注文のみに対応するため、注文のタイムスタンプは無視されることとなる。このことは、フロントランニングは対象取引の価格に影響を与えないが、取引が実行されるかどうかに影響を及ぼすことを意味する。

10 その他の攻撃

10.1 シビル攻撃と DOS 攻撃

悪意のあるユーザーは、ループリングノードを攻撃するために大量の小規模な注文を送信する可能性がある。しかし、ほとんどの場合において、ノードはこれらの注文をマッチングしても満足のいく利益を出せないことを理由にこれらの注文を拒否するであろう。中継器は自らどのように注文を取扱うのかを決める権限があるため、脅威として大量な小さな注文による攻撃は発生しない。

10.2 残高不足

悪意のあるユーザーは、実際のアドレスでは残高がゼロであるにも関わらず、注文数量がゼロでないことを偽った注文を署名し、送信する可能がある。その際、ノードは、実際の残高がゼロであることを監視して気付くことができる。状況に応じてこれらの注文状態を更新し、破棄することもできる。ノードは、注文のステータスを更新するために時間を費やさなければならないが、たとえば、アドレスをブラックリストに登録し、関連する注文を無効にすることなどもできる。

11 まとめ

ループリングプロトコルは、分散型取引所の基盤レイヤーとなることを目指しており。人々の資産や価値の交換方法を劇的に変えるものである。金銭は、中間財であり、取引の媒体として機能しており、取引の両者が互いの物品あるいはサービスを手に入れたい際の「欲求の二重の一致」(Double Coincidences of Wants) 問題 [21] を解決している。それと同様に、ループリングプロトコルは、取引をより簡単に完結させるためにリング・マッチングを使用することによって、取引ペアにおける欲求の二重の一致への依存性を取り除くことを目指している。これは、社会や市場がトークンや伝統的な資産をどのように交換するのかという点で大きな意味を持つ。実際に、分散型の仮想通貨は、国家の金銭に対する支配権を脅かしているのと同じように、トレーダー(消費者/生産者)を結びつけるプロトコルは、金銭の概念自体に対する理論的な脅威である。

ループリングプロトコルには以下のメリットがある。

- オフチェーンの注文管理とオンチェーンの決済によってパフォーマンスを低下させることなく、安全性を確保。
- リングマイニングと注文の共有による流動性の増大。
- 分散型取引場における大きな問題であるフロントランニングをデュアルオーサーリングで徹底的に防止。
- スマートコントラクトがオープンで無料であり、あらゆる dApp でプロトコルを使用可能。
- オペレータ間の標準化は、ネットワーク効果をもたらし、ユーザーエクスペリエンスを改善。
- オーダーブックとコミュニケーティングによるフレキシブルなネットワーク管理
- 参入敷居が低く、ノードがネットワーク及びエンドユーザーに参加するコストが低い。
- ウォレットによる直接的な匿名取引が可能。

12 謝辞

本書を作成するにあたり、メンター、アドバイザー、及びコミュニティーの皆様から、ご指導、ご協力をいただいた。ここに深謝の意を表する。特に、本プロジェクトにてレビューとフィードバックをしていただいた白碩(ChinaLedger より)、闞海濱教授、Alex Cheng、達鴻飛、曹寅、呉暁川、王震、于偉、段念、肖軍、銭

江、向江旭、郭一鵬、李大海、Kelvin Long、夏華夏、馬俊、及び Encephalo Path に感謝の意を表する。

参考文献

[1] Vitalik Buterin. Ethereum: a next generation smart contract and decentralized application platform (2013). *URL {http://ethereum. org/ethereum. html}*, 2017.

[2] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[3] Patrick Dai, Neil Mahi, Jordan Earls, and Alex Norta. Smart-contract value-transfer protocols on a distributed mobile application platform. *URL: https://qtum. org/uploads/files/cf6d69348ca50dd985b60425ccf282f3. pdf*, 2017.

[4] Viktor Atterlönn. A distributed ledger for gamification of pro-bono time, 2018.

[5] Hernando de Soto. *The Mystery Of Capital*. Basic Books, 2000.

[6] Fabian Schuh and Daniel Larimer. Bitshares 2.0: Financial smart contract platform, 2015.

[7] Bancor protocol. *URL https://bancor.network/*, 2017.

[8] Yaron Velner Loi Luu. Kybernetwork: A trustless decentralized exchange and payment service. `https://kr.kyber.network/assets/KyberNetworkWhitepaper.pdf`, Accessed: 2018-03-05.

[9] Fortune. How to steal $500 million in cryptocurrency. `http://fortune.com/2018/01/31/coincheck-hack-how`, Accessed: 2018-03-30.

[10] Robert McMillan. The inside story of mt. gox, bitcoin's 460 dollar million disaster. 2014.

[11] Sylvain Ribes. Chasing fake volume: a crypto-plague. `https://medium.com/@sylvainartplayribes/chasing-fake-volume-a-crypto-plague-ea1a3c1e0b5e`, Accessed: 2018-03-10.

[12] Rossella Agliardi and Ramazan Gençay. Hedging through a limit order book with varying liquidity. 2014.

[13] Will Warren and Amir Bandeali. 0x: An open protocol for decentralized exchange on the ethereum blockchain, 2017.

[14] Iddo Bentov and Lorenz Breidenbach. The cost of decentralization. `http://hackingdistributed.com/2017/08/13/cost-of-decent/`, Accessed: 2018-03-05.

[15] Daniel Wang. Coinport's implemenation of udom. `https://github.com/dong77/backcore/blob/master/coinex/coinex-backend/src/main/scala/com/coinport/coinex/markets/MarketManager.scala`, Accessed: 2018-03-05.

[16] Supersymmetry. Remarks on loopring. `https://docs.loopring.org/pdf/supersimmetry-loopring-remark.pdf`, Accessed: 2018-03-05.

[17] Fabian Vogelsteller. Erc: Token standard. *URL https://github.com /ethereum /EIPs /issues /20*, 2015.

[18] Chris Dannen. *Introducing Ethereum and Solidity*. Springer, 2017.

[19] Vitalik Buterin. Notes on blockchain governance. `https://vitalik.ca/general/2017/12/17/voting.html`, Accessed: 2018-03-05.

[20] Daniel Wang. Dual authoring —loopring's solution to front-running. *URL https://medium.com/loopring-protocol/dual-authoring-looprings-solution-to-front-running-d0fc9c348ef1*, 2018.

[21] Nick Szabo. Menger on money: right and wrong. `http://unenumerated.blogspot.ca/2006/06/menger-on-money-right-and-wrong.html`, Accessed: 2018-03-05.

付 録

付録 A　EVM(Ethereum Virtual Machine) 上のループリングプロトコルの実装

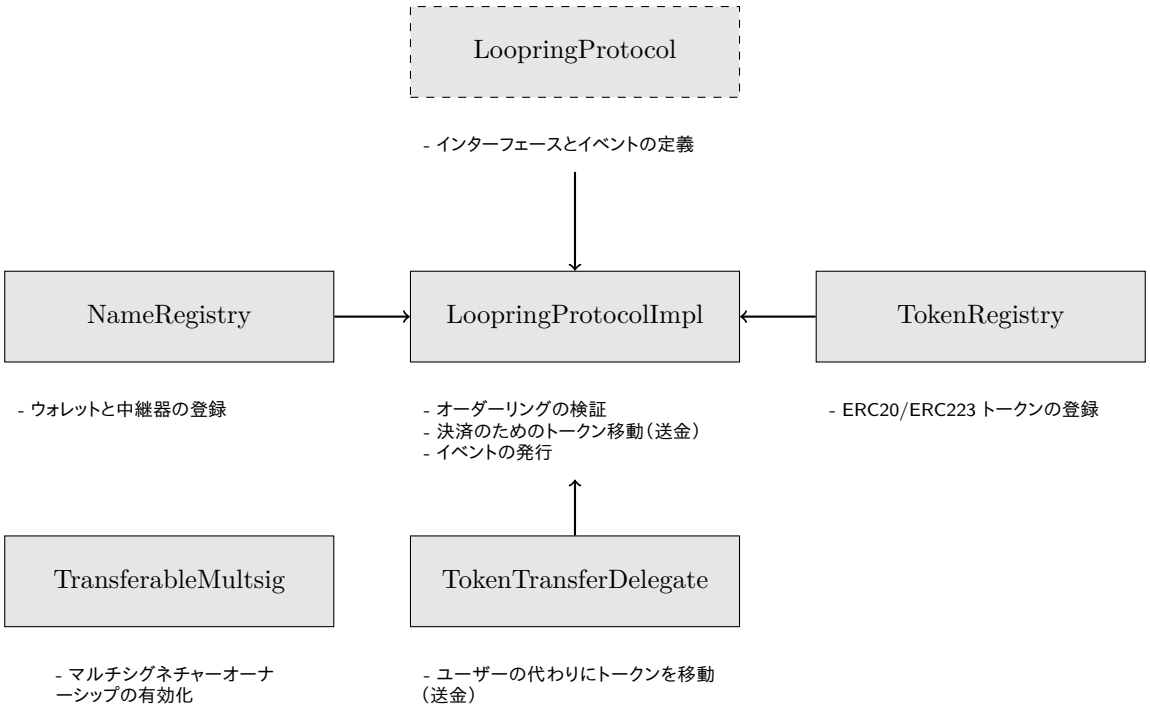

図 7: スマートコントラクト

付録 B　スマートコントラクトのデプロイメント

B.1　Ethereum

下記スマートコントラクトは既にイーサリアム (Ethereum) のメインネットにデプロイ済みである：

- LRC: `0xEF68e7C694F40c8202821eDF525dE3782458639f`
- TokenRegistry: `0xa21c1f2AE7f721aE77b1204A4f0811c642638da9`
- TokenTransferDelegate: `0x7b126ab811f278f288bf1d62d47334351dA20d1d`
- NameRegistry: `0xd181c1808e3f010F0F0aABc6Fe1bcE2025DB7Bb7`
- LoopringProtocolImpl: `0x0B48b747436f10c846696e889e66425e05CD740f`

B.2　Qtum

下記スマートコントラクトは既にクアンタム (Qtum) のメインネットにデプロイ済みである：

- LRQ:　`2eb2a66afd4e465fb06d8b71f30fb1b93e18788d`
- TokenRegistry:　`c89ea34360258917daf3655f8bec5550923509b3`
- TokenTransferDelegate:　`60b3fa7f461664e4dafb621a36ac2722cc680f10`
- NameRegistry:　`e26a27d92181069b25bc7283e03722f6ce7678bb`
- LoopringProtocolImpl:　`5180bb56b696d16635abd8dc235e0ee432abf25d`