LOOPRING（路印）：
去中心化代币交易撮合协议

王东 周杰 王辉
daniel@loopring.org jay@loopring.org alex@loopring.org
Matthew Finestone
matt.finestone@gmail.com

https://loopring.org

2018 年 5 月 9 日

摘要

路印是一种构建去中心化交易网络的开源协议，其不仅拥有一套公开的用于执行交易和撮合操作的智能合约，而且还有一组链下参与者用来撮合和广播订单。路印协议是免费、可扩展的，可以被看作是附带交易功能的去中心化应用（dApps）的标准构成要素。它的互操作标准促进了无需信任的匿名交易的实现。与当前现有的去中心化交易协议相比，路印带来的一个重要改进是能够将不同订单进行撮合、匹配，从而消除了两个代币交易对的约束，大大提高了流动性。此外，路印还部署了一套独特的防范抢先交易（一种不公平的，试图以比原始解决方案提供商更快的速度向区块提交交易的行为）的增强解决方案。另外，路印是区块链不可知的，可以部署到任何具备智能合约功能的区块链上。在本文撰写之时，路印协议已经在 Ethereum（以太坊）[1] [2] 和 Qtum（量子链）[3] 上成功运行，在 NEO [4] 上正在进行部署。

1 简介

随着区块链资产的激增，交易双方之间交换这些资产的需求大大增加。伴随着数以千计的新代币的出现——包括传统资产的代币化——这一需求又被放大了。无论是出于投机交易动机，还是想要通过交换来的原生应用代币实现接入网络的目的，对于更宏大的生态系统而言，实现加密数字货币资产之间的互换都是一项基本需求。事实上，这些资产 [5] 中存在着某种潜在的能量，将这种能力释放出来——解锁资本——不仅要求拥有资产所有权（区块链已经永久性地实现了这点），而且还需要让交易双方能够自由地转移和转换这些资产。

正因如此，无需信任的代币（价值）交易成为了区块链技术一个引人注目的使用案例。然而，直到如今，大多数加密数字货币爱好者依然满足于在传统的中心化交易所交易代币。因此市场对于路印协议的需要是迫切的，正如比特币 [6] 一直以来所强调的那样，对于点对点电子现金而言，“如果我们依然需要一个可信赖的第三方来防范‘双花’问题的发生，那么点对点现金对于我们而言就失去了优势”。同理，如果我们依然需要借助可信赖的、封闭的、中心化的交易所来交易去中心化资产，去中心化资产的显著特性也就不复存在了。

从哲学角度讲，在中心化交易所交易去中心化代币也根本说不通，因为其违背了这些去中心化项目所秉持的理念。此外，在下文中我们也详细描述了使用中心化交易所所面临的诸多风险和限制。一直以来，去中心化交易 (DEXs) [7] [8] [9] 也在试图解决这些问题，而且在许多情况下已经成功运用区块链技术去中介化的特点减轻了一些安全风险。然而，随着 DEX 的性能逐步成为新经济时代的重要基础设施，目前的 DEX 仍有很大的性能提升空间。路印的目的就是用其 dApp 不可知的开源协议为这些基础设施的构建提供模块化工具。

2 当前交易所现状

2.1 中心化交易所存在的不足

中心化交易所面临的三大主要风险是：1）缺乏安全性，2）缺乏透明度，3）缺乏流动性。

缺乏安全性源于用户常常都是把自己的私钥（资金）交给一个中心化的实体来管理。这种做法使用户不得不面临中心化交易所成为恶意黑客攻击牺牲品的可能。不过，尽管大家都知道所有中心化交易所都面临安全和被黑客攻击的风险 [10] [11] ，但却常常把这种风险看作是一种代币交易的“入场筹码”。目前，中心化交易所仍然是黑客攻击的“蜜罐”，因为他们的服务器上保管着超过数百万美元的用户资金。此外，交易所开发人员在用户资金管理方面也可能会犯一些诚实、偶然的错误。简单地说，用户将代币充值到一家中心化交易所之后，这些代币就不再受他们控制了。

缺乏透明度会让用户不公地面临交易所不正当行为所带来的风险，这里我们主要指的是交易所运营商的恶意行为。因为对于用户而言，实际上他们在中心化交易所交易的并不是自己的资产，而是 IOU。代币发送到交易所钱包之后，便开始由交易所保管，然后其会为用户提供 IOU 记账。因此，用户之后的所有交易实际上是这些 IOU 之间的交易。提现时，用户会向交易所申请赎回他们的 IOU，由交易所将相应代币发送至用户在交易所以往的钱包地址。整个过程缺乏透明，交易所有可能突然关闭，冻结你的帐户或者破产。此外，交易所还可能会将替用户保管的资产挪作他用，比如借给第三方。缺乏透明虽然可能不会造成用户损失全部资金，但有可能会带来较高的交易手续费，需求高峰期订单延迟，监管风险以及订单被抢先交易等风险。

缺乏流动性从交易所运营商的角度看，碎片化的流动性会抑制新交易所的出现，因为有可能两家交易所就可以吃掉整个市场。首先是拥有交易对最多的交易所会取胜，因为用户想要在一家交易所操作自己所有的交易。其次是订单表深度最厚的交易所，因为每个交易对都存在着可观的买卖价差。这种情况会抑制新来者加入竞争，因为它们难以建立起初始流动性。这种现象带来的结果是很多交易所尽管用户抱怨颇多，或者甚至发生过重大黑客攻击事件，但却依然占有着较高的市场份额。值得注意的是，随着中心化交易所占的市场份额越高，成为更大的攻击目标的可能性就越大。

从用户的角度看，碎片化的流动性会大大影响用户体验，因为在中心化交易所，他们只能在交易所自身的流动池、订单表以及所支持的交易对中进行交易。想要用代币 A 来交易代币 B，用户必须选择一家同时支持这两种代币交易的交易所，或者在不同的交易所提供自己的个人信息，注册帐号。此外，用户常常需要执行一步初始或者中间交易，多数是对 BTC 或 ETH 交易对，并在这一过程中支付买卖价差。最后，如果不下调订单量，用户有可能由于订单表深度不够而无法完成交易。即使交易所声称自身可以承受大额交易，但依然不可以保证这些成交量和流动性不是伪造的 [12]。

其结果是，流动性散落，生态系统支离破碎，类似于传统的金融体系，大量的交易量都集中在少数交易所。区块链的全球流动性承诺在中心化交易所中毫无价值。

2.2 去中心化交易所存在的不足

去中心化交易所与中心化交易所不同，部分原因是在去中心化交易所中，用户控制这他们的私钥（资产）在区块链上执行交易。通过利用加密货币自身无需信任的技术，它们成功地减少了许多上述提到的安全风险。然而，去中心化交易所依然存在着性能和结构限制等问题。

而且常常来讲，流动性依然是个问题，因为用户必须在不同的流动性资金池中寻找交易对手方。如果 DEX 或 dApp 不部署可互操作的统一标准，不能做到在一个更广的网络中分享/广播订单，流动性分割效应就依然存在。限价订单表的流动性，也就是弹性——被吃掉的限价单多快可以重新生成——可能会极大影响用户的最优交易策略 [13]。缺乏这样的标准不仅会降低流动性，而且还会让用户暴露于大量潜在不安全的专有智能合约的风险下。

此外，由于交易是在链上执行的，DEX 继承了底层区块链的限制，即可扩展性、执行延迟（挖矿）以及修改订单所需的高额费用。因此，区块链上的订单表扩展性并不是很好，因为在区块链上执行代码需支付费用（油费），使得高频取消多个订单变得昂贵得令人望而却步。

最后，由于区块链订单表是公开的，每笔交易都需要等待被矿工打包进下一区块，然后进入新订单表，矿工可以看到每笔发起的交易。这种延迟会使用户面临订单被抢先交易或最终成交价格/订单执行对用户不利的风险。

2.3 混合解决方案

基于上述原因，纯粹基于区块链技术的交易所有其局限性，在中心化交易所面前毫无竞争力。因此，我们需要在去中心化交易所链上固有的无需信任的特性和中心化交易所的交易速度和订单灵活性之间做出取舍。诸如路印和 0x [14] 等协议延伸出一种将链上撮合与链下订单管理相结合的解决方案。这些解决方案均围绕开放的智能合约而展开，然后又通过在链下执行几个函数，赋予节点灵活性以使其在整个网络扮演关键角色等措施克服了去中心化交易所的扩展性限制。然而，混合模型仍然存在缺陷 [15]。在本文中，路印协议在设计混合解决方案的方法上提出了一些有意义的改变。

3 路印协议

路印本身并不是一个 DEX，而是一个能够在多个不同区块链上构建 DEX 的模块化协议。我们把传统交易所的组成部分拆开来，用一套公开的智能合约和去中心化的参与者来替代。协议网络中的角色包括钱包、中继、流动性共享联盟链、订单表浏览器、环路矿工和资产代币化服务。在定义每个参与者之前，我们首先应该了解一下路印订单。

3.1 订单环路

我们用单向订单模式（UDOM）[16] 来表示路印订单。UDOM 把订单看作是代币交易请求 amountS/amountB（卖/买数量），而不是卖单和买单。由于每笔订单表示的只是两个代币间的兑换率，路印协议一个强大的特性是可以在订单环路中将不同的订单进行混合、匹配。借助于高达 16 种不同的订单类型，而不是一个单一的交易对，路印协议可以极大提高流动性以及价格增长潜力。

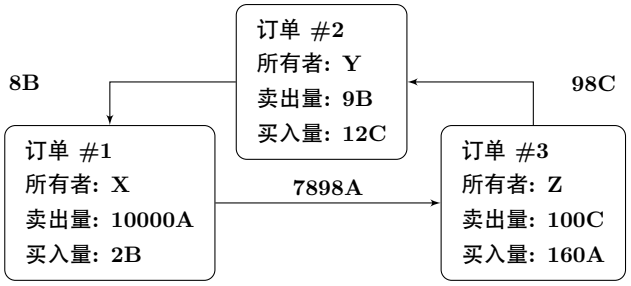

图 1: 一个由 3 笔订单组成的环路

上图显示了一个由 3 笔订单组成的订单环路。每笔订单想要卖出的代币 (tokenS) 都是另一笔订单想要买入的代币 (tokenB)。由此产生的一个环路允许订单之前相互交换他们想要的代币，并不需要一个与之相对的订单与其构成交易对。当然，传统的订单对交易依然可以执行，只不过本质上讲，这种情况在订单环路中属于特例。

定义 3.1 (订单环路) *如果有 n 个不同的代币 C_0, C_1, $\cdots$, C_{n-1} 和 n 个订单 $O_{0\to1}$, $\cdots$, $O_{i\to i\oplus1}$, $\cdots$, $O_{n-1\to0}$，那么这些订单可以组成跨 n 个代币种的交易环路:*

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

其中 n 为环路的长度，$i \oplus 1 \equiv i+1 \mod n$

当所有组成部分的交易都能以等于或者优于用户隐式指定的初始兑换率而执行时，订单环路便是有效的。要验证订单环路的有效性，路印协议智能合约必须接收来自矿工的订单环路，其中所有订单的原始兑换率乘积必须等于或大于 1。

我们假设 Alice 和 Bob 想交易他们的代币 A 和 B。Alice 有 15 个代币 A，想用其买入 4 个代币 B；Bob 有 10 个代币 B，想用其买入 30 个代币 A。

那么，到底谁在买，谁在卖? 这只取决于我们所确定的提供定价参考的资产。如果代币 A 是定价参考，那么 Alice 就是在以 $\frac{15}{4} = 3.75$A 的价格买入代币 B，而 Bob 则是以 $\frac{30}{10} = 3.00$A 的价格卖出 10 个代币 B。如果代币 B 是定价参考，那么 Alice 就是在以 $\frac{4}{15} = 0.26666667$B 的价格卖出 15 个代币 A，Bob 在以 $\frac{10}{30} = 0.33333334$B 的价格买入 10 个代币 A。因此，买方或者卖方都是任意的。

在第一种情况下，Alice 愿意支付比 Bob 的代币卖价 (3.00A) 更高的价格 (3.75A)，而在第二种情况下，Bob 愿意支付比 Alice 的代币卖价 (0.26666667B) 更高的价格 (0.33333334B)。很明显，只要买方愿意支付的价格等于或高于卖方价格，就有可能进行交易。

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

因此，要使一个由 n 个订单组成的订单集合能够完全或部分成交，我们需要知道每笔订单的买单兑换率汇率乘积是否大于或等于 1。如果是的话，所有 n 个订单就可以部分或全部成交 [17]。

如果我们引入一个第三方 Charlie，使订单满足 Alice 想要花费 x_1 个代币 A 买入 y_1 个代币 B，Bob 想要花费

x_2 个代币 B 买入 y_2 个代币 C，Charlie 想要花费 x_3 个代币 C 买入 y_3 个代币 A 的条件，所有必需的代币都已经到位，那么交易在满足下列条件时是可能的：

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

欲了解更有有关路印订单的细节，请参考 7.1 部分。

4 生态系统参与者

以下生态系统参与者共同提供了中心化交易所必须提供的所有功能。

- **钱包**：一种常见的钱包服务或接口，不仅可以使用户获取代币，而且也是一种向路印网络发送订单的方式。钱包可以在与环路矿工共享手续费的激励下，不断产生新的订单 (详见第 8 部分)。我们相信未来的交易将在个人用户钱包的安全范围内进行，通过我们的协议将这些流动性资金池连接起来是至关重要的。

- **流动性共享联盟链/中继网络**：一种用于订单及流动性分享的中继网络。当节点运行路印中继软件时，它们能够加入现有网络，通过联盟链与其他中继共享流动性。我们正在构建的有史以来的第一个联盟链，不仅可以实现几乎实时共享订单 (1-2 秒区块时间)，而且还可以通过对旧有交易历史进行微调以允许新节点更快地下载订单。值得注意的是，中继不必加入这个联盟；它们可以单独行动，不与他人共享流动性，或者可以创建和管理自己的流动性共享网络。

- **中继/交易撮合**：中继是从钱包或中继网络接收订单，维护公共订单表和交易历史，以及选择性地向其他中继（通过任何任意的链下介质）和/或中继网络节点广播订单的节点。交易撮合是中继的一种特性，而不是一种要求，其计算量大，完全在链下运行。我们把拥有交易撮合特性的中继称为 “RingMiners”，它们通过将不同的订单拼接在一起来生成订单环路。中继在以下方面是自由的：(1) 如何选择与其他中继进行交流；(2) 如何构建订单表；以及 (3) 如何挖掘订单环路（挖矿算法）。

- **路印协议智能合约 (LPSC)**：一套公开而免费的智能合约，可以用来检验从矿工处接收到的订单环路，代表用户无需信任地结算和转账代币，用手续费激励矿工和钱包，以及发布事件。中继/订单浏览器负责监听这些事件，以保持它们的订单和交易历史是最新的。详情见附录 **??**。

- **资产代币化服务 (ATS)**：为那些无法在路印协议上直接交易的资产架起一座桥梁，是一项由可信赖的公司或组织管理的中心化服务。用户将资产（实物、法币或其他链上的代币）充入之后将得到发行的代币，可在将来赎回。虽然路印并不是一种跨链交易协议（直到有合适的解决方案为止），但 ATS 使得 ERC20 代币 [18] 与实物资产以及其他区块链资产之间的交易成为了可能。

5 交易流程

1. **协议授权**：在图 2 中，需要交易代币的用户 Y 先对路印协议智能合约 (LPSC) 进行授权，使其能够转出用户想要销售的代币 B（amountS）。该操作不会冻结用户的代币，换句话说，在订单执行过程当中，用户依然能够自由转移自己的代币。

2. **创建订单**：类似于订单表浏览器的中继或其他中介负责提供代币 B 和代币 C 之间的兑换率与订单表。用户 Y 可以借助任一完整的钱包界面定义卖出量 amountS 、买入量 amountB 以及其他参数，从而发起一个订单（限价订单）。发起过程中会有一定数量的 LRx 作为环路矿工的矿工费被加入订单，LRx 的费用越高，订单就能越快地被环路矿工执行。用户 Y 会用自己的私钥对订单的哈希值进行签名。

3. **订单广播**：钱包会将订单以及其数字签名发送到一个或多个中继。而后，中继会更新他们的公共订单表。对于订单表的架构原则，如先到者先服务，该协议本身并不作要求。因此，在构建订单表时，各网络中继享有自由设计的权力。

4. **流动性共享**：中继可以通过任意通信媒介对订单进行广播，这也再次说明各节点之间能够灵活地交流。为方便建立稳定的网络连接，联盟链上部署了一个流动性共享中继网。就像之前章节中所提到的，该中继网能够加快交易速度并且优化订单的参与度。

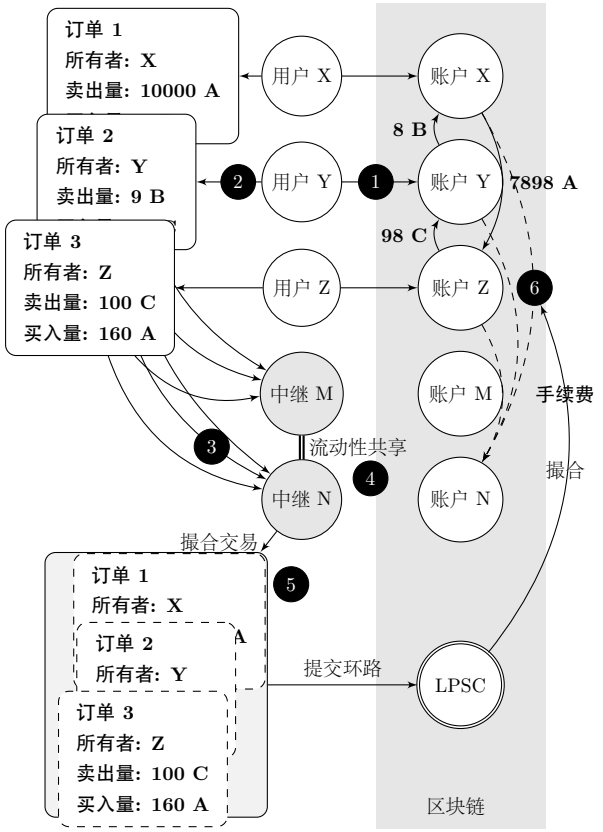

图 2: 路印协议交易流程

5. **环路挖矿（订单撮合）**：环路矿工会通过撮合多个不同的订单，尽力按照用户给定或者更好的兑换率促使订单达成完全或部分成交。环路撮合是路印协议能够为任何交易对提供高流动性的主要原因。如果订单实际执行的兑换率优于用户 Y 指定的兑换率，利润部分由该环路中的所有订单共享。作为回报，环路矿工可以选择抽取该利润的一部分（利润分割，并且将作为矿工费的 LRx 退还给用户）或仅仅收取 LRx 费。

6. **验证与结算**：LPSC 接收到环路订单之后会进行多次检查，以验证矿工提供的数据，决定该订单环路是否可以完全或部分撮合（取决于环路中订单的成交率以及用户钱包中的代币余额）。如果所有检查都是成功的，LPSC 会自动将代币转给用户，同时支付给矿工和钱包相应的费用。如果 LPSC 判定用户 Y 的钱包余额不足时，交易量会按比例缩减：与取消订单（整个过程是单向的人工操作且无法撤销）不同的是，等到足够的代币存入钱包地址后，缩减后的订单会自动增加至初始订单量。

6 操作灵活性

值得注意的是，路印协议的开放标准赋予了参与者极大的操作灵活性。每个参与者都能自由采取新型商业模式，为用户带来价值，并在这一过程中根据成交量或其他衡量标准赚取一定量的 LRx 手续费（如果他们这样选择的话）。整个生态系统是模块化的，旨在激励更多应用参与其中。

6.1 订单表

中继在设计订单表时，可以以任何方式展示和匹配用户的订单。目前我们的第一个订单表采用的 OTC 模式，限价订单只根据价格进行排序。换句话说，订单的时间戳顺序对订单表没有任何影响。不过，中继在设计自己的订单表时，可以自由选择订单排列方式，模仿典型的中心化交易的撮合引擎，在尊重时间戳的同时按照价格对订单进行排序。如果一个中继倾向于选择这种类型的订单表，他们可以选择对接一款钱包，使得钱包将订单单独发送给某一个中继，这样一来，该中继便能够以时间顺序撮合订单。在路印协议中，任何这样的配置都是可能的。

与其他去中心化交易协议有时会要求中继拥有一定的资源——用来下买单的初始代币余额——不同，路印协议中的中继只需要找到可撮合的订单来完成一笔交易，即便没有初始代币也能做到。

6.2 流动性共享

中继可以自由设定彼此之间共享流动性（订单）的方式。我们的联盟链只是实现流动性共享的一种方案，整个生态系统中的成员可以根据偏好，自由地交换信息。除了加入联盟链之外，只要他们觉得合适，便可以构建和管理自己的流动性共享网络，制定相关的规则/激励措施。当然，就像之前在钱包实现方案中看到的以时间顺序排序订单的方案一样，中继也可以单独工作。虽然选择其他中继交流合作对于追求网络效率而言有着明显的优势，但不同的商业模式依然可以给设计独特的分享机制带来价值，按照任何方式分享手续费。

7 协议详解

7.1 订单分解

订单就是一组描述用户交易意图的数据。我们用单项订单模式（UDOM）来表示路印订单，具体如下：

```
message Order {
  address protocol;
  address owner;
  address tokenS;
  address tokenB;
  uint256 amountS;
  uint256 amountB;
  unit256 lrcFee
  unit256 validSince; // 系统时间
  unit256 validUntil; // 系统时间
  uint8   marginSplitPercentage;  // [1-100]
  bool    buyNoMoreThanAmountB;
  uint256 walletId;
  // 双重授权地址
  address authAddr;
  // v, r, s组成了签名
  uint8   v;
  bytes32 r;
  bytes32 s;
  // 双重授权私钥,
  // 不用来计算订单哈希值,
  // 因此未被签名.
  string  authKey;
  uint256 nonce;
}
```

为了保证订单来源，一笔订单是由用户的私钥对除 `authAddr` 以外的参数哈希进行签名。`authAddr` 参数用来对订单所在的订单环路进行签名，从而防范抢先交易（欲了解更多细节，请参考 9.1 部分）。该签名由 `v`, `r`, `s` 字段表示，并与订单参数一起发送至网络。此举确保了一笔订单在整个生命周期内都是不可更改的。尽管如此，路印协议依然可以根据订单地址余额以及其他变量来计算出订单的当前状态。

UDOM 不包括价格（本质上必须是浮点数），而是使用术语 `rate` 或 r，用 `amountS`/`amountB` 来表示。该兑换率不是一个浮点数，而是一个表达式，只有在需要时才会使用其他无符号整数进行计算，以将所有中间结果保持为无符号整数，从而提高计算精度。

7.1.1 买入量

当环路矿工撮合订单时，有可能会发现更好的可执行兑换率，让用户获得比指定买入量（`amountB`）更多的 `tokenB`。然而，如果 `buyNoMoreThanAmountB` 函数设置为 `True`，协议便会确保用户收到不超过 `amountB` 的 `tokenB`。因此，UDOM 中的 `buyNoMoreThantokenB` 函数决定了一笔订单完全成交的时间。`buyNoMoreThantokenB` 参数对 `amountS` 或 `amountB` 设定了上限，相比于传统的买卖订单，它也使用户能够更加细节化地表达自己的交易意向。

例如，当 `amountS` $= 10$ ，`amountB` $= 2$ 时，兑换率为 $r = 10/2 = 5$。因此，用户愿意卖出 5 个 `tokenS` 来换取一个代币 `tokenB`。环路矿工撮合时为用户找到了更优的兑换率 4，使得用户能收到 2.5 个而不是 2 个 `tokenB`。然而，如果用户只想买入 2 个 `tokenB`，并且将 `buyNoMoreThanAmountB` 设置为 `True`，LPSC 按照兑换率 4 执行交易，用户以 4 个 `tokenS` 买入 1 个 `tokenB`，实际上最终节约了 2 个 `tokenS`。请记住，这里尚未考虑矿工费 (详见 8.1 部分)。

实际上，如果我们用

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

命令来表示一个简化形式的订单，那么对于传统交易所中的 ETH/USD 交易市场来说，传统的买卖模式可以用下面的第 1 个和第 3 个订单来表达，但其他 2 个订单却不可以：

1. 以 300USD/ETH 的价格卖出 10 个 ETH，可以被表示成：`Order(10, ETH, 3000, USD, False)`.
2. 以 300USD/ETH 的价格卖出 10 个 ETH，得到 3000USD。这可以被表示成:`Order(10, ETH, 3000, USD, T`
3. 在 300USD/ETH 的价格买入 10 个 ETH，可以被表示成：`Order(3000, USD, 10, ETH, True)`.
4. 花 3000USD 在 300USD/ETH 的价格买入尽可能多的 ETH，这可以被表示成：`Order(3000, USD, 10, ETH, False)`.

7.2 环路验证

LPSC 并不负责执行计算兑换率或交易量操作，而是必须接受环路矿工提供的这些值，然后对其进行验证。之所以这些计算要由环路矿工来完成，有以下两个主要原因：（1）智能合约的编程语言，比如以太网网络中的 solidity[19] 语言，不支持浮点数计算，尤其是工作量证明机制 $pow(x, 1/n)$（计算一个浮点数的 n 次方根）(2) 为减少区块链上的计算和成本，计算过程最好在链下完成。

7.2.1 子环路检验

这一步是为了防止套利者通过在订单环路中执行新订单来不正当地获取所有利润。本质上，只要环路矿工发现了一个合法的订单环路，就有可能会产生向订单环路中添加其他订单以攫取用户全部的利润（兑换率折价）的想法。如下图 3 所示，只要仔细计算 $x1$, $y1$, $x2$ 和 $y2$，使环路中所有订单的兑换率乘积正好等于 1，就没有兑换率折价了。

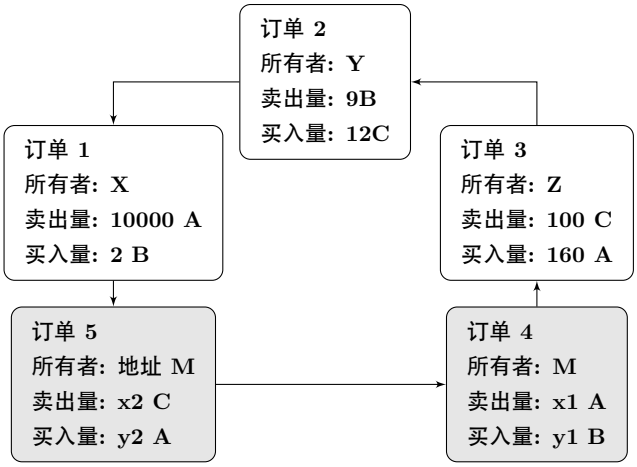

图 3: 含有子环路的环路订单

虽然这种做法对于整个网络是零风险的，但却不会带来任何价值，我们将其看作是环路矿工的不正当行为。为了防止这种行为的出现，路印协议规定一个合法的环路内不能有任何子环路。为了确保这点，LPSC 会确保一个代币不能买入或卖出两次。在上图中，我们可以看到代币 A 作为卖出代币和买入代币各被交易两次，这一点是不被允许的。

7.2.2 成交率检验

我们已经在上文中阐述了订单环路中的兑换率计算必须有环路矿工执行的原因，即必须由 LPSC 来验证正确性。首先，LPSC 会验证矿工对于每笔订单所执行的买入率等于或小于用户设定的买入率，从而确保用户能在交易中至少得到他们要求的乃至更优的兑换率。其次，一旦兑换率确定之后，LPSC 会确保订单环路中的每个订单都享有相同的兑换率折价比例。例如，如果兑换率折价比例为 γ，那么每个订单的价格将会是：

$$r_{0\to1}\cdot(1-\gamma),\ r_{1\to2}\cdot(1-\gamma),\ r_{2\to0}\cdot(1-\gamma),\ \text{并且满足：}$$

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

因此：

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

因此，如果整个交易中包含 n 个订单，`discount`（折价）是：

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

r^i 表示第 i 个订单的周转率。显然，只有当折价比例 $\gamma \geq 0$ 时，这些订单才会成交；第 i 个订单 (O^i) 的实际兑换率为：$\hat{r^i} = r^i\cdot(1-\gamma)$, $\hat{r^i} \leq r^i$.

回顾我们之前的例子，Alice 拥有 15 个代币 A 并想用其买入 4 个代币 B，Bob 拥有 10 个代币 B 并想用其买入 30 个代币 A。如果代币 A 是定价参考，那么 Alice 就是在以 $\frac{15}{4} = 3.75$A 的价格买入代币 B。Bob 就是在以 $\frac{30}{10} = 3.00$A 的价格卖出代币 B。折价为：$\frac{150}{120} = 1.25$ 也就是 $\frac{1}{1.25} = 0.8 = (1--\gamma)^2$。因此在实际交易中对双方都公平的兑换率为 $\sqrt{0.8}\cdot 3.75 \approx 3.3541$ 代币 A /代币 B.

Bob 给出 4 个代币 B 收到了 13.4164 个代币 A，多于他为这 4 个代币 B 所预期的 12 个代币 A。Alice 收到了 4 个代币 B 但只给出了 13.4164 个代币 A 少于她所期望的 15 个代币 A。注意，一部分利润会被当作手续费来奖励给矿工（和钱包）。(详见 8.1 部分)。

7.2.3 成交追踪与取消

用户可以通过向 LPSC 发送一笔特殊交易部分或全部取消一个订单。该交易需要包括订单的详细内容以及计划取消的数量。在这之后，LPSC 会将这笔交易收录，保存将要取消的数量，发布一个 `OrderCancelled` 事件至网络中。LPSC 会利用订单哈希值作为标识符，存储他们的值，不断追踪成交与取消的数量。这些数据是可以公开访问的，当数据发生变化时，会发出 `OrderCancelled` / `OrderFilled` 事件。对 LPSC 来说，在环撮合阶段追踪这些值是至关重要的。

此外，LPSC 也同样支持使用 `OrdersCancelled` 事件取消用户地址下某一交易对所有订单，以及 `AllOrdersCancelled` 事件取消用户地址下所有交易对订单。

7.2.4 订单缩减

根据历史成交量、取消量以及发送者账户中的当前余额，订单可以被缩减。整个过程需要根据上面提到的订单特点，找到买入/卖出量最小的订单，然后将其作为在订单环路中对所有交易量进行缩减的参考。

找到最小订单可以帮助我们计算出每个订单的成交量。比如，如果第 i 个订单是最小订单，那么每个订单中卖出的代币数量 $\hat{s}$ 与每个订单中买入的代币数量 $\hat{b}$ 可以通过如下计算得出：

$$
\begin{aligned}
&\hat{s}^i = \overline{s}_i, \hat{b}^i = \hat{s}^i / \hat{r}^i, ; \\
&\hat{s}^{i\oplus 1} = \hat{b}^i, \hat{b}^{i\oplus 1} = \hat{s}^{i\oplus 1} / \hat{r}^{i\oplus 1}; \\
&\hat{s}^{i\oplus 2} = \hat{b}^{i\oplus 1}, \hat{b}^{i\oplus 2} = \hat{s}^{i\oplus 2} / \hat{r}^{i\oplus 2}; \\
&...
\end{aligned}
$$

其中，$\overline{s}_i$ 指的是订单部分成交后的中的账户余额。

在实际操作中，我们可以放心地假设环路中的任意订单都具有最小订单，然后订单在环路中不断循环，通过最多两次计算就能得到每个订单的成交量。

例如，如果可以成交的最小数量是原始订单的 5%，订单环路中的所有交易量便会缩减到 5%。当交易完成时，之前被认为是待成交数量最小的订单应该已经完全成交了。

7.3 环路撮合

如果订单环路通过了之前的所有检验，那么该订单环路便可以关闭，交易完成。这意味着所有 n 个订单形成了一个关闭的订单环路，如图 4 所示：

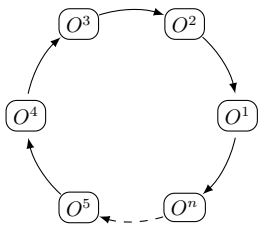

图 4: 环路撮合

为了完成交易，LPSC 会调用 `TokenTransferDelegate`（代币转让代理）智能合约。引入这样一种代理机制使得升级协议智能合约变得更加简单，因为所有订单只需授权该代理，而不是协议的所有不同版本。

对于订单环路中的每个订单，根据执行情况，将会有一定数量的 `tokenS` 支付给下一个或上一个订单。然后，根据矿工选择的收费模式支付矿工费。最后，一旦所有交易全部完成，便会发布一个 `RingMined`（撮合完成）事件。

7.3.1 发布事件

路印协议发布事件，允许中继、订单浏览器和其他参与者尽可能高效地接收订单更新。发布的事件有：

- **OrderCancelled**：某一特定订单已被取消。
- **OrdersCancelled**：用户地址下某一交易对所有订单已全部取消。
- **AllOrdersCancelled**：用户地址下所有交易对订单已全部取消。
- **RingMined**：某一订单环路已经成功撮合。该事件包含了环路内每笔代币转账的详细数据。

8 LRx 代币

LRx 是路印统一的代币符号。LRC 是在以太坊网络上的路印代币，LRQ 是在量子链网络上的路印代币，LRN 是在 NEO 网络上的路印代币。随着路印不断在其他公有链上进行部署，未来我们还会引入其他类型的 LRx 代币。

8.1 收费模式

当用户创建订单时，他们会指定一定数量的 LRx 用来作为支付给矿工的手续费，一同连带的还有矿工能从该笔订单中分得的利润比例 (`marginSplitPercentage`)。我们将其称之为分润比例。选择哪一种（手续费或分润比例）方式完全由矿工自主决定。

下图是一种分润比例展示图：

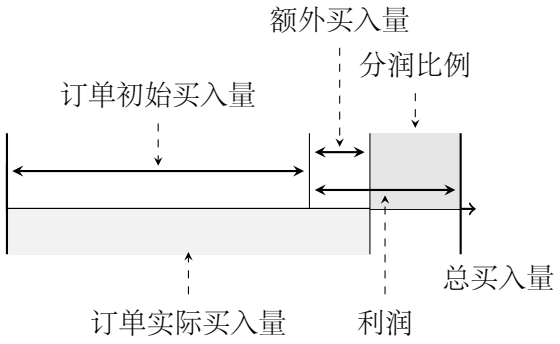

图 5: 一种 60% 的分润比例

如果订单环路上的利润率太小，矿工将会选择 LRx 模式。相反，如果利润率足够大，以致于由此产生的分润比例远远超过 LRx 费用，则矿工会选择分润比例模式。然而，这里有另一个限制条件：当矿工选择分润比例模式时，他们必须向用户（订单创建者）支付一笔费用，数量相当于用户本应支付给矿工的 LRx 费用。这就将矿工选择分润比例模式的门槛提高到 LRx 费用模式的两倍，增强了选择 LRx 模式的倾向。这使得矿工能够在低利润率的环路订单上获得固定收入，以换取在较高利润率的环路订单上获得较少收入的平衡。对于这种收费模式，我们希望随着市场的发展和成熟，利润率较高的环路订单环将会减少，因此需要固定的 LRx 费用作为激励。

我们最终会得到下列图表：

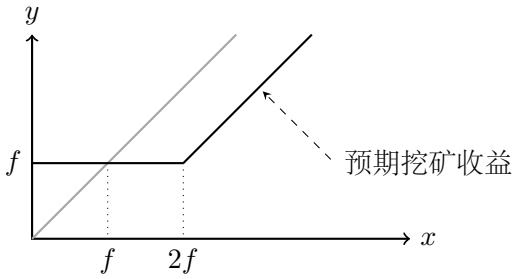

图 6: 路印收费模式

其中 f 代表 LRx 费用，x 代表分润比例，y 代表挖矿收入。实线表示 $y = max(f, x - f)$；如果一笔订单的 LRx 费用为 0，则用 $y = max(0, x - 0)$ 方程式表示。简单来讲，就是灰色实线代表 $y = x$。

由此带来的结果是：

1. 如果分润比例为 0，矿工将会选择没什么吸引力的 LRx 费用模式，依然获得奖励。
2. 如果 LRx 费用为 0，便会出现灰色实线。此时矿工的收入将将呈一般线性模型。
3. 当分润比例收入大于 LRx 费用模式的两倍时，矿工会选择分润比例模式，支付 LRx 给用户。

需要注意的是，如果 LRx 费用不为零，不论矿工选择哪种费用模式，矿工与订单发起者之前总会发生 LRx 转账，矿工要么选择挣 LRx 费，要么就要将 LRx 费返还给订单发起者，然后收取分润比例费。

此外，矿工还将与钱包分享一定比例的费用。当用户通过一个钱包发起一笔订单并且最终成交时，会有一定比例的费用或分润比例奖励给钱包。尽管这种设置是模块化的，但我们依然可以根据需要设计独特的商业模式或应用，我们倾向于让钱包从矿工处中大约分得 20%-25% 的收益。因为钱包虽然是集成路印协议的主要目标，但其自身却只有很少或没有任何收入来源。

8.2 去中心化自治

从根本上讲，路印协议是一种社会协议，它依赖于成员之间的相互协作，从而有效实现某一目标。虽然这与一般密码经济协议并无不同，但实际上，它的有用性很大程度上受到同样的协调问题机制 [20]、严峻的触发平衡和有限理性的保护。为此，LRx 代币的用途不仅仅只是支付费用，而且还用于协调各个网络参与者的财政激励。这种协调对于实现任何协议的广泛采用都是必要的，而且对于交换协议来说尤为重要，因为路印协议最终的成功与否在很大程度上取决于能否在一个健全的去中心化生态系统中提高流动性。

LRx 代币将用于通过去中心化自治实现协议更新。未来，智能合约的更新将由代币持有者负责治理，从而确保连续性、安全性，并通过不兼容性降低流动性流失风险。考虑到智能合约一经部署便不可修改，dApp 或终端用户可能会面临继续与被废弃的版本交互，且无法使用更新版协议的风险。升级能力是协议成功的关键，因为它必须适应市场和底层区块链的发展需要。LRx 股东的去中心化治理将允许协议智能合约在不影响 dApp、终端用户，或过度依赖智能合约提取的的情况下实现升级。LRx 代币发行总量是固定的，以 LRC 为例，一定比例的代币归路印基金会所有，一定比例的代币被用作支持社区发展的资金。

然而，LRx 代币持有者并不是唯一的掌握路印协议发展方向的股东：中继/环路矿工、钱包、开发人员和其他参与者也是整个生态系统的一个组成部分，路印协议的

发展也必须听取他们的意见和建议。事实上，鉴于这些参与者不需要持有任何 LRx 代币来履行各自的角色（因为路印协议中不存在传统的卖方/买方和做市商角色，初始代币准备金并不是强制性的），我们必须通过其他方法来满足他们的利益。此外，“简单”的以代币为基础的投票，无论是在链上还是在链下，都是不完美的，会带来分歧，因为低投票率和代币过渡集中会带来风险。因此，我们的目标是实现一个构建在层中的治理模型，做到在达成共识的基础上建立一套标准决策过程。我们可以借助协调机构来收集各个参与者群体，也许还可以是预先确定的协议联络点的不同声音。随着这一成果的实现，路印基金会必然会逐步从协议开发人员演变为协议管理员。

9 欺诈和攻击保护

9.1 防范抢先交易

在去中心化交易中，抢先交易指有人试图复制另一节点的交易方案，撮合未成交交易池（mempool）已外的交易，可以通过指定较高的交易费用（油费）来实现。在路印协议中存在的抢先交易（以及其他订单撮合协议）主要为订单偷窃：抢先交易者从未成交的环路撮合交易中偷取一个或多个订单。具体到路印便是：抢先交易者从未成交交易中偷取整个订单环路。

当提交的环路交易（submitRing）没有被确认并且仍然在未成交交易池中时，这样的交易很容易被人发现，环路偷窃者可以轻而易举地把自己的地址 (the `filcherAddress`) 替换成矿工地址（`minerAddress`）对交易负载重新签名，接着 `filcherAddress` 就取代了订单环路签名。偷窃者可以设置更高的油费，提交一笔新的交易，吸引矿工优先撮合他提交的新交易，而不是原来提交的环路交易。

面对这一问题，我们之前提出的解决方案存在一些重要缺陷：它需要更多的交易支持，因此矿工会消耗更多的油费，同时撮合一个环路要花费至少双倍的区块时间。我们的新的“双重授权”（Dual Authoring）解决方案包括为订单设置两个级别的授权机制：一个用于结算，另一个用于挖矿。

双重授权流程：

1. 对于每个订单，钱包软件将生成随机的公钥/私钥对，并将密钥对加入订单的 JSON 片段中。（另一种方法是使用公钥派生的地址代替公钥本身，以减小字节大小，我们使用 `authAddr` 来表示这样一个派生地址，`authKey` 表示 `authAddr` 的配对私钥）。
2. 用订单中除 `r`, `v`, `s`, 和 `authKey` 以外所有的字段计算出订单的哈希值，然后使用订单所有者的的私钥（不是 `authKey`）进行签名。
3. 钱包会将订单和 `authKey` 一起发送给中继进行撮合。矿工将验证 `authKey` 和 `authAddr` 是否正确配对，且订单的签名对订单所有者地址有效。
4. 当发现一个新提交的订单环路时，矿工将使用每个订单的 `authKey` 对环路哈希、`minerAddress` 和所有挖矿参数进行签名。如果一个环路包含 n 个订单，`authKeys` 就将产生 n 个签名，我们称这些签名为 `authSignatures`。矿工也许还需要使用 `minerAddress` 的私钥对环路哈希及所有挖矿参数进行签名。
5. 矿工使用所有参数和额外的 `authSignatures` 调用 submitRing 函数。请注意，`authKeys` 不是链上交易的一部分，因此除矿工本身没有任何人知道。
6. 路印协议现在会对每个订单相应的 `authAddr` 验证其 `authSignature`，如果发现任何丢失或无效的 `authSignature`，就会拒绝该环路的交易。

由此带来的结果是：

- 订单的签名（通过订单所有者地址私钥签署）保证订单和 `authAddr` 不被修改。
- 矿工的签名（由 `minerAddress` 的私钥签署），如果提供，保证没有人可以窃取他/她的身份进行环路撮合。
- `authSignatures` 保证整个环路都不能被篡改，包括 `minerAddress`，因此订单是无法被窃取的。

双重授权可以防止环路偷窃和订单偷窃，同时确保订单撮合可以在一个交易中完成。另外，双重授权还为中继的订单共享提供了两种方式：不匹配共享和可匹配共享。默认情况下，路印的运作模式属于 OTC，并且只支持限价订单，这意味着订单的时间戳将被忽略，意味着抢先交易虽然确实会影响交易的执行与否，但对交易的实际成交价格没有任何影响。

10 其他攻击

10.1 Sybil 或 DOS 攻击

恶意用户——以自身身份或伪造身份——可以通过发送大量小额订单来攻击路印节点。然而，由于我们允许节点根据自身标准拒绝订单——它们可能会选择隐藏或公开这些标准——大多数这样的订单将因为匹配时不能产生令人满意的利润而被拒绝。通过授权中继来决定如何管理订单，我们认为大规模的小订单攻击并不是一种威胁。

10.2 余额不足订单攻击

恶意用户可以对订单值为非零但其地址余额实际为零的订单进行签名和广播。节点监视并注意到某些订单的实际余额为零后，相应地更新这些订单状态，然后丢弃它们。虽然节点必须花费时间更新订单的状态，但也可以选择通过例如将地址列入黑名单和删除相关订单的方式将工作量最小化。

11 总结

路印协议致力于成为去中心化交易服务的基础层。这一过程将会对人们如何交易资产和价值产生深刻影响。货币作为一种中间商品，促进或替代了物物交换，解决了“双方需求的巧合匹配”问题 [21]，即双方必须给予彼此不同的商品或服务。同样，路印协议旨在通过使用环路撮合来更容易地完成交易，从而消除我们对交易对中需求重合的依赖。这对于社会和市场如何交易代币/传统资产以及其他更广泛的事物都很有意义。事实上，正如去中心化加密数字货币对国家对货币的控制构成威胁一样，一种能够大规模撮合交易者（消费者/生产者）的组合协议，理论上也会对货币概念本身构成威胁。

路印协议带来的好处包括：

- 链下订单管理与链上撮合结合在一起意味着没有因为安全考虑而牺牲任何性能。
- 环路撮合和订单分享带来了更大的流动性。
- 双重授权方案解决了当今所有 DEX 和其用户所面临的极为有害的抢先交易问题。
- 免费而公开的智能合约使任何 dApp 都可以在路印协议上构建应用或与协议本身进行交互。
- 运营商之间形成的统一标准提高了网络效率和终端用户体验。
- 在管理订单表和广播订单时的网络灵活性更强。
- 降低准入门槛意味着节点加入网络和终端用户所需的费用更低。
- 实现了用户钱包间直接的匿名交易。

12 致谢

在这里，我们想由衷地感谢我们的导师、顾问以及广大路印社区参与者，感谢他们如此热情和慷慨地用自身学识对白皮书中的内容提出意见。在此，我们要尤其感谢中国分布式总账基础协议联盟（ChinaLedger）白硕；阚海滨博士；Alex Cheng，达鸿飞；曹寅；吴小川；Zhen Wang、于伟、段念、肖军、钱江、向江旭、郭一鹏、李大海、Kelvin Long、夏华夏、马俊和 Encephalo Path 对本项目的评审建议和意见。此外，我们也要在此感谢付金涛、张弛、时晓飞三位译者对于本白皮书的中文翻译所付出的努力。

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] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

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

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

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

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

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

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

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

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

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

[16] 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.

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

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

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

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

[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.