# 0x ：以太坊上开放的去中心化交易所协议

Will Warren, Amir Bandeali

0xProject.com

2017-2-21

# 摘要

我们描述了一个协议，用来降低在太坊链上ERC20代币的点对点的交易摩擦。该协议目的旨在开放标准和通用构建方式，推动包含交易能力的去中心应用(dApps)之间的交互性。这些由以太坊智能合约执行的交易是可公开访问的，免费使用，并且所有的dApp都可以与其挂钩。建立在这套协议上的DApps都可以访问公共的流动币池或者创建自己的流动币池，最终根据交易量收取费用。这套协议是开明的：它不会对在用户施加成本，也不会牺牲组用户的价值让另一组用户收益。去中心化的管理是持续的，并且安全地集成更新升级，不会影响到dApps和用户。

# 1. 介绍

区块链充满了变革，它允许所有人通过一个开放的金融网络拥有和转移资产，而不需要可信赖的第三方。现今有数百种基于区块链的资产[1]，并且每个月都有新的种类在增加，这些资产的交易需求在剧增。随着智能合约出现，各种区块链资产不需要可信赖第三方的相互交易成了可能。

从中心化到去中心化交易生态系统是一个重要进展，主要有几点原因：去中心化可以给用户提供强大的安全保障，因为不会有卷款逃走，或被黑客攻击和被政府监听的中央机构。Mt.Gox, Shapeshift, BitFinex[2,3]被黑客攻击证明了中心化交易里这种风险是很明显的。去中心化将会消除这些风险，不使用中间机构也让用户放心交易，同时让安全监管放在所有用户上而不是单个托管人。

过去两年, 从以太坊创世块诞生开始，各种去中心的应用已经建立了各种智能合约来实现点对点的交易。快速迭代以及缺乏实践指导导致区块链的应用实现十分分散，缺乏整合。因此，用户必须面对各种实现了同样功能的智能合约，但是这些智能合约都拥有各自的配置和学习曲线，这种状态增加了网络的不必要损耗，将用户隔离在自己使用的特定应用中，破坏了价值网络效应的流动性。

0x 是一个用在以太坊上的开放的去中心化交易所协议。它旨在作为一个基础构建，可以和其他各种协议进行组合用于驱动日益复杂的dApps[4]。0x 使用公开的智能合约系统，可以作为各种dApp的共享基础架构。如图1所示。在长时间的运行中，开放的技术标准往往胜于封闭的。并且随着越来越多的资产成为链上的代币，我们会看到更多需要使用不同代币的dApps。因此，需要一个开放的交易标准去支持这种开放的经济。

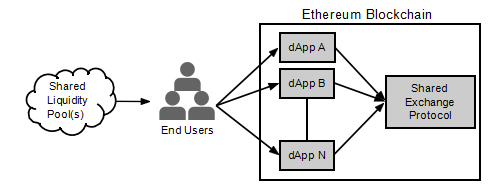


图1：开放的协议应该与应用无关的。将协议实现与应用实现之间进行解耦分离，就像开发者和使用者之间一样。

# 2. 现状

由于以太坊智能合同开发的去中心式交易的设计效率低下，对做市商造成较高的摩擦成本而未能产生巨大的交易量。特别是如果要他们的订单放在链上[5-8]，那得要求做市商在每次发布，修改或取消订单时花费gas。虽然一笔交易的成本很小，但是频繁修改订单以应对市场变化就会使成本累计得很高。除了给做市商带来高成本之外，维持链上的订单将导致交易消耗网络带宽，使区块链膨胀却不一定带来价值转移。

自动做市商（Automated market maker ， AMM）智能合同[9,10]被提出作为替代链上订单的方式。AMM智能合同取代订单，采用价格调整模式，其中资产的现货价格会强烈响应于AMM市场交易中任意一方的市场力量和市场参与者。AMM的优点包括可用性（可以经常作为交易对手，虽然提供的现货价格可能比从更传统的交易所获得的更差），并且方便于与需要外部执行市场订单的智能合同整合。价格调整模型的确定性使其对市场流动性不敏感，这意味着交易导致价格在厚薄市场[11]中调整相同的数量。换句话说，AMM对供给曲线施加人为约束。如果价格调整模式过于敏感，即使是小额交易也会导致现货价格的大幅波动。如果价格调整模式不够敏感，AMM的资金将很快被套利者耗尽。

状态通道（state channels）被建议用在于缩减以太坊链的规模，并通过将交易移出区块链[13,14]来降低各种应用包括交换[12]的成本。状态通道的参与者来回传递加密签名的消息，累积中间状态更改，而不将其发布到正式链，直到通道关闭。状态通道对于“条形码”应用是理想的，其中许多中间状态改变可以在通过单个链上交易（即日间交易，扑克，回合制游戏）结算之前在链外累积。如果其中一个通道参与者离开通道或尝试作弊，则有一个挑战时期，在此期间，其他参与者可以发布他们从犯罪人那里收到的最新消息。因此，通道参与者必须始终在线，挑战不诚实的交易对手，导致参与者容易遭受DDOS攻击。虽然状态通道大大减少了特定用例的链上交易数量，但开放和安全关闭状态通道所需的许多在线交易和安全存款繁琐的步骤使其对一次性交易效率低下。

一个混合型的实现，我们称之为“链外订单中继与链上结算”（off-chain order relay with on-chain settlement），将状态通道的效率与即时结算的链上订单相结合。在这种方法中，加密签名的订单从链外广播; 感兴趣的交易对手可以将一个或多个这些订单注入一个智能合约，直接在链上无条件地执行交易[15,16]。对于做市商而言摩擦成本可以最小化，因为它们可以在链外签名，交易只有在价值转移时才会发生。 我们通过允许任何人成为交易所，并让协议与应用分离来拓展这种方法。

# 3. 说明书

图2为链外订单中继和链上结算的一般顺序。现在我们暂时忽略稍后会变得重要的几个机制

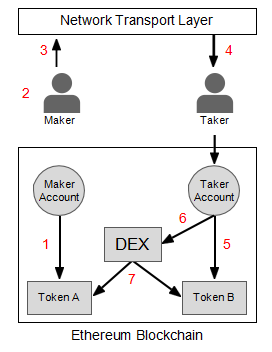


图2：链外订单中继，链上结算图。灰色矩形和圆圈分别代表以太坊智能合同和帐户。指向以太坊智能合同的箭头表示函数调用; 箭头从调用方指向被调用方。智能合同可以在其他智能合同中调用功能。以太坊附带的箭头代表信息流。

1. maker同意去中心交易（DEX）合约，以获取代币A的余额。
2. maker创建代币A替换代币B的订单，指定所需的汇率，到期时间（超过该时间无法填写订单），并使用其私钥对订单进行签名。
3. maker通过任意媒体渠道进行广播订单。
4. taker接收到订单，并决定他们想填单。
5. taker同意DEX合同访问代币B的余额。
6. taker提交maker签名的订单到DEX合同。
7. DEX合同认证maker的签名，验证订单尚未到期，验证订单尚未被填单，然后以指定汇率转让双方之间的代币。

## 3.1 消息格式 Message Format

每个订单都是包含订单参数和关联签名的数据包。订单参数通过Keccak SHA3散列加密为32个字节。订单创建者使用其私钥对订单散列进行签名以产生ECDSA签名。

### 3.1.1 点对点订单 Point-to-point Orders

点对点订单允许交易双方使用他们喜欢的媒体渠道直接交换代币。构成订单的数据包是可以通过电子邮件，Facebook消息，whisper （一款社交软件）或以类似服务发送的十六进制几百字节大小得数据。 订单只能由指定的Taker地址填充，使得该订单对窃听者或外部用户无效。

表1： 点对点订单的数据格式

|  |  |  |
| --- | --- | --- |
| **字段** | **数据类型** | **描述** |
| Version | address | 交易所智能合约的地址，当协议改变时，地址也会变 |
| maker | address | 生成订单的地址 |
| taker | address | 允许填单的地址 |
| tokenA | address | 代币的合约地址 |
| tokenB | address | 代币的合约地址 |
| valueA | uint256 | 代币A的总额 |
| valueB | uint256 | Maker请求的代币B额度 |
| expiration | uint256 | 订单过期时间(unix秒) |
| v | uint8 | 签名生成的参数 |
| r | bytes32 |
| s | bytes32 |

### 3.1.2 广播订单 Broadcast Orders

对于流动性市场的出现，必须有一个公共平台，其中买方和卖方可以发布订单，随后汇总成订单簿，即交易所。建立和经营交易所的成本很高，目前为止我们描述的协议并不能鼓励人们承担这种费用。广播订单通过允许任何人作为交易所，维护订单（公共或私人）并对所有流动性收取交易费用来解决这个问题。我们指的是托管和维护订单的Relayer，而不是交易所。在交易所必须建立和运营专有基础设施，执行交易并处理用户资金的情况，Relayer仅通过托管和传播由通用消息组成的订单簿来促进市场参与者之间的代币交易。Relayer不代表市场参与者执行交易，因为这需要市场参与者信任Relayer。相反，taker执行自己的交易。

广播订单的消息格式给点对点消息格式的做出了两点更改，以促进公众交换和激励Relayer。首先，广播订单不指定taker地址，允许任何人拦截和填写广播订单。其次，广播订单包括指定交易的费用A，费用B和费用接收者三个参数，指定了具体交易费用和用来收取交易费用的Relayer地址。交易所智能合约将这些费用在填写订单时转移到费用接收者。图3显示了maker和Relayer用来协商交易费用的步骤顺序。

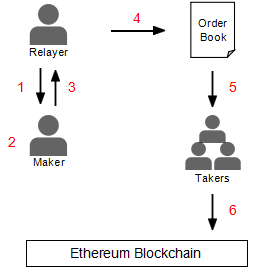


图3：relayer主持和维护一个链外订单，以换取交易费用。 该图显示了链外订单中，maker和relayer以无需信任的方式谈判交易费用的步骤顺序。 在交易结算时，交易费用从maker或taker又或者两者同时转移到relayer，从而扩充了图2所示的链上结算流程。

1. relayer引用了费用表及用于收取交易费用的地址。
2. maker创建一个订单，将feeA和feeB设置为满足relayer的费用表的数值，设置feeRecipient为relayer期望的接收地址，并用他们的私钥对订单进行签名。
3. maker将签名的订单发送给relayer。
4. relayer收到订单，检查订单是否有效以及提供的费用。 如果订单无效或不符合relayer的要求，则拒绝订单。 如果订单合格的话，relayer将订单更新到订单簿中。
5. taker收到包含maker订单的订单簿的更新版本。
6. taker通过发送到以太坊链上的交易智能合约，将填好的maker的订单。

表2: 广播订单的消息格式

|  |  |  |
| --- | --- | --- |
| **字段** | **数据类型** | **描述** |
| version | address | 交易所智能合约的地址，当协议改变时，地址也会变 |
| maker | address | 生成订单的地址 |
| tokenA | address | 允许填单的地址 |
| tokenB | address | 代币的合约地址 |
| valueA | uint256 | 代币的合约地址 |
| valueB | uint256 | 代币A的总额 |
| expiration | uint256 | Maker请求的代币B额度 |
| feeRecipient | address | relayer的地址 |
| feeA | uint256 | maker需要支付给relayer的协议代币. |
| feeB | uint256 | taker需要支付给relayer的协议代币. |
| v | uint8 | 签名生成的参数 |
| r | bytes32 |
| s | bytes32 |

尽管maker指定交易费用似乎很奇怪，但请记住，relayer最终可以控制哪些订单被发布。因此，如果maker希望将订单发布到特定订单簿，那么必须将feeA，feeB和feeRecipient设置为满足与该订单簿相关联的relayer的值。由于费用时在链外协商的，relayer可能会动态更改收费表，并自行决定（对于尚未签署的订单，而不是现有订单）。relayer可以使用链上或链外的信息来设置和调整费用，允许灵活的费用表（固定费用，基于百分比，基于体积，分层，订阅模式等）。但是，一旦relayer接受了订单，订单的费用值就无法改变。

传统的交易所服务使用的匹配引擎来代表用户填单，用户必须相信交易所将为他们提供最优惠的价格。通常，用户可以放心，如果这些受监管实体试图作弊，或匹配引擎发生了故障，它们将被追究责任。因为0x协议保持的无信任机制，relayer将不能代表maker和taker执行交易。相反，relayer只能向taker推荐最佳价格，taker自主决定是否签署和发送交易到区块链。这意味着0x协议不能支持真正的市场订单，但是，设计良好的Web应用程序可以提供这种类型的用户体验。

重要的是要认识到，feeRecipient可以指向任何智能合约。这意味着复杂的relayer结构可以“加入”0x协议。例如，可以根据每个节点在审查性p2p网络中传播订单的贡献水平，将feeRecipient合约设计为在多个relayers或一系列节点中分配交易费用。

## 3.2 智能合约

交易所协议是在可以公开访问和免费使用的以太坊智能合同中实施的（不会对用户征收超出标准gas成本的费用）。 它以Solidity编程语言编写，包含两个相对简单的功能：填单和取消(fill and cancel)。 整个合同约为100行代码，并且大约需要90k的gas来填单。

### 3.2.1 签名认证 Signature Authentication

交易所智能合约能够使用ecrecover函数来验证订单发起者（maker's）的签名，该函数将哈希和哈希签名作为参数返回产生签名的公钥。 如果ecrecover返回的公钥等于maker地址，则签名是真实的。

address publicKey = ecrecover( hash, signature( hash ) );

if ( publicKey != maker ) throw;

### 3.2.2 填单和部分填单 Fills & Partial Fills

交易所智能合约能存储对每个先前填单的引用，以防止单次订单被多次填充。这些引用存储在映射中; 在这种情况下，将32字节的数据块会映射成256位无符号整数。 将与订单相关联的参数传递到Keccak SHA3函数中会产生唯一的32字节散列，这一散列可以用于唯一标识该顺序（哈希冲突的几率，找到具有相同散列的两个不同阶数得可能性，几乎为零）。 每次填单时，映射都存储订单散列和填单的累计值。

在调用交易所智能合约的填充函数时，taker可以通过指定附加参数valueFill来部分填充订单。 只要部分填充的总和不超过订单的总值，就可以在单个订单上执行多个部分填充。

表3:当意图填写订单时 taker必须提供额外的参数

|  |  |  |
| --- | --- | --- |
| **字段** | **数据类型** | **描述** |
| valueFill | uint256 | 总共需要填的tokenA的值 (valueFill <= valueA). |

### 3.2.3 过期时间 Expiration Time

订单的到期时间由maker在订单签署时指定。到期时间是一个无符号整数值，表示自unix时期以来的绝对秒数。签名后无法更改此值。

以太坊虚拟机中的时间由块时间戳给出，每个时间戳在每次开始新的块时设置。因此，订单的到期状态并不取决于taker广播他们意图填单的时间，而是取决于矿工在EVM中执行填单功能的时间。矿工不能将当前块的块时间戳设置为早于先前块的时间戳。

### 3.2.4 取消订单 Cancelling Orders

通过交易所智能合同的取消功能，相关的maker可能会取消未填写同时未过期的订单。取消功能将订单的散列映射到订单的最大值（valueA），防止后续填写。取消订单需要花费gas，因此，取消功能仅用作回退机制。通常，maker应该通过设置其订单过期时间和打算更新其订单的频率相匹配以避免链上交易。

这种方法的一个问题是，它会制造一种情况：maker尝试取消一个订单，同时taker试图填写一个相同的订单。其中一方的交易将会失败，导致浪费gas，这取决于两个交易的挖矿顺序。有关挖矿交易序列的不确定性可能会导致不良后果。如果以太坊遇到大量的待处理交易积压，这种不确定性可能会增加。

# 4.协议代币 Protocol Token

密码经济协议创造了财政激励措施，推动理性经济主体网络协调其行为，以完成一个过程[4,17,18]。而0x基本上是一种网络协议，用于促进买方和卖方之间的代币（而不是一个加密经济协议），它旨在作为包含交易功能的dApps的开放标准。建立和维护开放标准是一个协调问题，为参与方增加了运营开销;当各方有不同的需求和经济激励的时候，协调就会变得难以处理。协议代币可以协调财务奖励和抵消与围绕单一技术标准组织多方相关的成本。即使调整采用激励措施是有用的，但协议代币可用于解决更具挑战性的问题：在未来，成为不可变的智能合同系统中去中心化治理的协议。

## 4.1 去中心化治理 Decentralized Governance

### 4.1.1 持续集成 Continuous Integration

一旦将以太坊智能合约部署到区块链上，其内部逻辑就无法改变。因此，为了更新协议，必须部署一个全新的智能合约，该合约要么网络分叉，要么干扰依赖该协议的用户和进程，直到他们“选择加入”最新版本为止。在交易所中，破坏性协议更新可能使所有未结算订单无效，并要求每个市场参与者批准新的智能合同来获取其交易余额。或者，该协议可以划分成并行运行的两个版本，减缓因dApp互操作性创建的网络效应。虽然智能合同约的抽象可能被用于将不间断地更新集成到协议中而不会中断更高级别的进程，但是这种更新机制也可能为最终用户带来重大的安全风险（在最坏的情况下，攻击者可以获得用户资金的访问权）。协议代币可用于驱动去中心式更新机制，允许将更新集成到协议中，同时也保护协议的用户和利益相关者。

0x将被部署到具有固定供应的协议代币的以太坊链上，这些代币将被发布给合作伙伴的dApps和将来的终端用户。协议代币将有两个用途：市场参与者向relayer支付交易费用，和对协议的更新进行去中心治理。根据图4所示的过程，去中心治理将用于安全地更 新集成到0x协议。最初，一个简单的多签名合同将用于去中心治理，直到开发出更复杂的DAO。 0x协议及其原生代币不会对用户带来不必要的成本，从relayer处寻求租金或提取价值。协议的智能合约将被公开访问并完全免费使用。没有机制可以使一个群体牺牲让另一个群体受益。

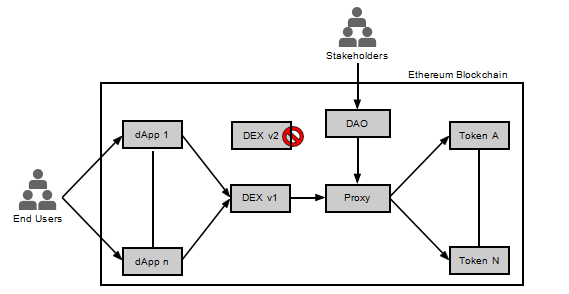


图4：可以通过合约抽象和分散式治理的组合方式部署协议更新，而不需要中断网络。终端用户提供代理合同，以访问他们计划用于交易的代币。利益相关方提出并选择通过DAO在全新的智能合同（DEX v2）中实施的协议改进。 DAO授权新的智能合约访问用户代币，将其添加到代理合约的白名单中，最终取消列出已弃用的协议版本。

### 4.1.2 代币注册 Token Registry

订单由机器可读的十六进制字节码组成，但并不一定适合人类阅读。代币注册合同将用于存储每个代币的相关元数据的ERC20代币列表，其中包括这些元数据：名称，符号，合同地址和表示代币最小单位所需的小数位数（需要确定汇率）。注册管理机构将作为官方的链上参考，市场参与者可以在执行交易之前独立地验证代币地址和汇率。由于代币注册表将作为可靠信息来源，因此将需要监督从注册表添加，修改或删除代币的过程。 0x利益相关者将提供这种监督。虽然代币注册表可以方便用户验证其订单的完整性，但是0x协议可以被使用来交易任何使用ERC20代币接口的代币。

在将来，协议的订单格式可以被修改以便于人的阅读。代币可以通过在代币注册表中注册的三个字符符号来标识，而不是由代币的合同地址来标识。以太坊域名服务（ENS）可用于通过人类可读的名称（例如“theDunkle.eth”）来识别maker，taker和relayer，而不是通过帐户或合同地址。

# 5.总结

* 链外订单中继+链上结算=做市商的低摩擦成本+快速结算。
* 任何dApp都可以挂钩公开访问的智能合约。
* Relayers可以创建自己的流动资金池并收取交易费用。
* 标准化+解耦=共享协议层→
* 提供dApps之间的互操作性
* 创造互惠互利的流动性网络效应
* 减少进入门槛，降低市场参与者的成本
* 消除冗余，提高用户体验和智能合同安全
* 去中心更新机制可使改进能够连续，安全地集成到协议中而不会中断dApps或用户。

# 6.致谢

我们要感谢我们的导师，顾问和Ethereum社区的许多人，他们非常热情和愿意分享知识。 特别是，我们要感谢Joey Krug，Linda Xie和Fred Ehrsam，对这项工作进行审查，编辑和提供反馈。 我们还要感谢我们在Silicon Valley Ethereum Meetup遇到的组织者和社区成员，包括Joseph Chow，Martin Koppelmann，Rebecca Migirov，Gustav Simonsson，Grant Hummer，Tom Ding和String Labs等人。

# 7.附录

## 7.1 ERC20代币

ERC20为代币的组成建立了标准合同ABI，并已成为所有类型数字资产的实际代表。ERC20代币共享相同的合约接口，简化了与外部合约的集成。

核心ERC20功能包括：

* transfer（to, value）
* balanceOf（owner）
* approve（spender，value）
* allowance（owner，spender）
* transferFrom（from，to，value）

EIP101包含了一个用于改变以太坊使其符合ERC20代币标准的提议。现在，可以使用“包装”智能合同作为ERC20以太网的代理。有关参考，请参阅Maker实现或Gnosis实现。

## 7.2 合约 ABI

EIP50建议扩展合同ABI以支持结构。这将允许社区建立标准的订单和签名数据结构，简化我们的合约界面和与外部合约的集成。

## 7.3 以太坊域名服务

EIP137或Ethereum名称服务（ENS）将可阅读的名称（例如“my- name.eth”）解析为可能代表Ethereum地址，Swarm和/或IPFS内容散列或其他标识符的机器可读标识符。 它也可以用于将元数据与名称相关联，例如合同ABI或whois信息。 ENS将被0x协议使用，以创建更直观的消息格式以根据名称引用maker，taker和relayer。

# 参考文献

[1]coinmarketcap. https://coinmarketcap.com/all/views/all/. Accessed: 2017-02-016.

[2]Wikipedia: Mt. Gox. https://en.wikipedia.org/wiki/Mt.Gox. Accessed: 2017-02-016.

[3]A Timeline: ShapeShift Hacking Incident. https://info.shapeshift.io/blog/2016/04/19/timeline- shapeshift-hacking-incident. Accessed: 2017-02-016.

[4]Will Warren. The difference between App Coins and Protocol Tokens. https://medium.com/@willwarren89, 2017.

[5]Maker Market. https://mkr.market/. Accessed: 2017-02-01.

[6]EtherOpt. https://etheropt.github.io/. Accessed: 2017-02-01.

[7]Augur. https://augur-dev.firebaseapp.com/. Accessed: 2017-02-01.

[8]Intrinsically Tradable Tokens. https://www.reddit.com/r/ethereum/... Accessed: 2017-02-01.

[9]Euler. https://www.reddit.com/r/ethereum/... Accessed: 2017-02-01.

[10]Galia Benartzi Guy Benartzi, Eyal Hertzog. Bancor protocol: A hierarchical monetary system and the foundation of a global decentralized autonomous exchange. 2017.

[11]Abraham Othman, David M Pennock, Daniel M Reeves, and Tuomas Sandholm. A practical liquidity-sensitive automated market maker. ACM Transactions on Economics and Computation, 1(3):14, 2013.

[12]RaidEX. http://www.raidex.io/. Accessed: 2017-02-014.

[13]Jeff Coleman. State Channels. http://www.jeffcoleman.ca/state-channels/. Accessed: 2017-02-014.

[14]Ledger Labs: State Channels Wiki. https://github.com/ledgerlabs/state-channels/wiki. Accessed: 2017-02-014.

[15]IDEX, Decentralized Capital. http://www.idex.market/. Accessed: 2017-02-01.

[16]EtherDelta. https://etherdelta.github.io/. Accessed: 2017-02-01.

[17]Fred Ehrsam. App Coins and the dawn of the Decentralized Business Model. https://blog.coinbase.com, 2016.

[18]Fred Ehrsam. How to Raise Money on a Blockchain with a Token. https://blog.gdax.com, 2016.