Uniswap V2 核心白皮书

- 翻译: 崔棉大师

摘要

该技术白皮书解释了Uniswapv2核心合约背后的一些设计决策。它涵盖了合约的新功能-包括任意ERC20之间的配对，强化的价格预言系统（允许其他合约估计给定时间间隔内的时间加权平均价格），"快速交换"（flash swaps），交易员可以在接收资产之前在其他地方使用这些资产，然后再付款，以及将来可以开启的协议费。它还会重新设计合约以减少攻击面。本白皮书描述了Uniswap v2"核心"合约的机制，包括存储流动性提供者资金的配对合约以及用于实例化配对合约的工厂合约。

1.简介

Uniswap v1是在以太坊区块链上的智能合约的链上系统，基于"不变产品公式" [1]实现了自动流动性协议。每个Uniswap v1配对都存储两个资产的合并储备，并为这两个资产提供流动性，从而保持了储备产品不能减少的不变性。交易者为交易支付30个基点的费用，该费用由流动性提供者支付。合约不可升级。

Uniswap v2是基于相同公式的新实现，具有一些非常令人希望的新功能。最重要的是，它支持创建任意ERC20/ERC20交易对，而不是仅支持ERC20和ETH之间的配对。它还提供了一个标准化的价格预言机，该价格预言在每个块的开头累加了两种资产的相对价格。这使得以太坊上的其他合约可以在任意间隔内估算这两种资产的时间加权平均价格。最后，它启用了"快速交换"功能，用户可以自由接收资产并在链上的其他地方使用它们，而仅在交易结束时支付或返还这些资产。

虽然合约通常不能升级，但是有一个私钥可以更新工厂合约中的变量以开启交易中的5个基点的链上费用。最初将关闭此费用，但将来可能会启用，此后，流动性提供者将在每笔交易中获得25个基点，而不是30个基点。

Uniswap v2还解决了Uniswap v1的一些小问题，并重新实现，减少了Uniswap的攻击面，并通过最小化持有流动性提供者的"核心"合约中的逻辑，使系统更易于升级。

实际使用Uniswap v2将需要通过"路由器"合约调用配对合约，该合约计算交易或存款金额并将资金转移到配对合约。

2.新功能

2.1 ERC-20

对Uniswap v1使用ETH作为过渡货币。每对都将ETH作为其资产之一。这使路由更简单（ABC和XYZ之间的每笔交易都通过ETH/ABC对和ETH/XYZ对），并减少了流动性的分散。但是，该规则对流动性提供者施加了巨大的成本。所有流动性提供者都有以太坊的敞口，并且由于其他资产相对于以太坊的价格变化而遭受永久损失。当两个资产ABC和XYZ相关时（例如，如果它们都是美元稳定币），Uniswap对上的流动性提供者ABC/XYZ所承受的永久损失通常会比ABC/ETH或XYZ/ETH对更少。

使用ETH作为强制性过渡货币也会给交易者带来成本。交易者必须支付的费用是直接购买ABC/XYZ对的费用的两倍，并且遭受两次滑点。Uniswap v2允许流动性提供商为任意两个ERC-20创建对合约。

任意ERC-20之间的配对数量激增，可能会更难找到交易特定配对的最佳路径，但路由可以在更高的层上通过链外或通过链上路由器或聚合器进行处理 。

2.2价格预言

可以通过将资产a的准备金除以资产b的准备金来计算Uniswap在时间t提供的边际价格（不包括费用）。

$$p_t = \frac{r_t^a}{r_t^b}$$

由于如果此价格不正确（套利金额足以弥补费用），套利者将与Uniswap进行交易，因此Uniswap提供的价格倾向于跟踪资产的相对市场价格，如Angeris所示[2]。意味着它可以用作近似价格的预言。

但是，Uniswap v1不能安全用作链上价格预言，因为它非常易于操作。假设其他一些合约使用当前的ETH-DAI价格结算衍生产品。希望操纵所测价格的攻击者可以从ETH-DAI对中购买ETH，触发衍生合约的结算（导致其根据虚高价格进行结算），然后将ETH卖回该对以进行交易(1)。甚至可以作为原子交易完成，也可以由控制区块内交易顺序的矿工完成。

Uniswap v2通过测量和记录每个块的第一次交易之前的价格来改进此预言机功能。这个价格比起区块价格要难操纵。如果攻击者提交的交易试图在区块末尾操纵价格，则其他套利者可能能够提交另一笔交易，随后立即在同一区块中进行交易。矿工或使用足够的gas来填满整个区块的攻击者可以在区块末尾操纵价格，但是除非他们也开采下一个区块，否则他们在套利交易方面可能没有特别的优势。

具体来说，Uniswap v2通过在有人与合约进行交互的每个区块的开头跟踪价格的累积总和来累计此价格。根据区块时间戳2，每个价格均按自上次更新区块以来经过的时间加权。(2)意味着在任何给定时间（更新后）的累加器值应为该价格的总和。合约历史上每秒的现货价格。

$$a_t = \sum_{i=1}^{t} p_i$$

为了估算从时间t_1到t_2的时间加权平均价格，外部可以在t_1处检查累加器的值，然后在t_2处再次检查，将第二个值减去第一个值，然后除以经过的秒数。（请注意，合约本身并不存储该累加器的历史值，调用者必须在时段开始时调用合约以读取和存储该值。）

$$p_{t1,t2} = \frac{\sum_{i=t_1}^{t_2} p_i}{t_2 - t_1} = \frac{\sum_{i=1}^{t_2} p_i - \sum_{i=1}^{t_1} p_i}{t_2 - t_1} = \frac{a_{t_2} - a_{t_1}}{t_2 - t_1}$$

预言机用户可以选择何时开始和结束此时间段。选择较长的时间段会使攻击者操纵TWAP的成本更高，尽管这会导致价格降低。

一种复杂情况：我们应该以资产B来衡量资产A的价格，还是以资产A来衡量资产B的价格？尽管以B表示的A的现货价格始终是以B表示的B的现货价格的倒数，但是在特定时间段内，资产A相对于资产B的平均价格不等于B的现货价格的倒数。资产B相对于资产A的平均价格(3)。

例如，如果美元/ETH价格在第1区块中为100，在第2区块中为300，则平均USD/ETH价格将为200USD/ETH，但平均ETH/美元价格将是1/150 ETH/USD。由于合约无法知道用户要使用两个资产中的哪个作为帐户单位，因此Uniswap v2会跟踪两个价格。

另一个复杂之处是，有人可以将资产发送到配对合约，从而更改其余额和边际价格，而无需与之交互，因此也不会触发预言机更新。如果合约只是简单地检查自己的余额并根据当前价格更新了预言机，则攻击者可以通过在首次调用该资产之前立即向合约发送资产来操纵预言机。

为防止这种情况，核心合约在每次交互后都会缓存其储备，并使用从缓存的储备而非当前储备中得出的价格来更新预言。除了保护预言机不受操纵外，此更改还可以使合约重新架构。

(1) 有关将Uniswap v1用作预言机如何使合约容易受到此类攻击的真实示例，请参阅[3]。
(2) 由于矿工有一定的自由来设置区块时间戳，因此预言机的用户应注意，这些值可能与实际时间不完全对应。
(3) 给定期间内资产A的资产A的算术平均价格等于该期间内资产B的资产B谐波平均价格的倒数。如果合约衡量的是几何平均价格，那么价格将是彼此的倒数。但是，几何平均TWAP不太常用，并且很难在以太坊上计算。

2.2.1 精确

由于Solidity不提供对非整数数值数据类型的支持，因此Uniswap v2使用简单的二进制定点格式来编码和处理价格，具体而言，给定时刻的价格存储为UQ112.112数字，这意味着在小数点的两侧指定了112个小数位精度，没有符号。这些数字的范围为$[0，\ 2^{112}-1]^4$(4)，精度为$\frac{1}{2^{112}}$ 。

出于实用性的原因选择了UQ112.112格式-因为这些数字可以存储在uint224中，这使256位存储插槽中的32位可用。还可能发生的情况是，每个存储在uint112中的保留空间在（打包的）256位存储插槽中还剩下32位可用空间。这些可用空间用于上述累加过程。具体而言，备用存储区与最近一次具有至少一笔交易的块的时间戳一起存储，并使用2^{32}进行修改，以使其适合32位。另外，尽管保证任何给定时刻的价格（存储为UQ112.112数字）都适合224位，但该价格在一定间隔内的累积是不正确的。存储插槽末尾的A/B和B/A累计价格额外的32位用于存储重复价格累加产生的溢出位。这种设计意味着价格预言仅在每个区块的第一笔交易中增加了额外的3个SSTORE操作（当前成本约为15,000gas）。

主要缺点是32位不足以存储不会合理溢出的时间戳记值。实际上，Unix时间戳溢出uint32的日期为02/07/2106。为了确保该系统在此日期之后以及以后的每$2^{32}-1$秒的倍数内继续正常运行，仅要求预言机每个间隔（约136年）至少检查一次价格。这是因为累加（和修改时间戳）的核心方法实际上是溢出安全的，这意味着考虑到Oracle正在使用正确的（简单）溢出算法来计算增量，可以适当考虑跨交易的溢出间隔。

(4) 理论上的$2^{112}-\left(\frac{1}{2^{112}}\right)$上限不适用于此设置，因为Uniswap中的UQ112.112数始终由两个uint112的比率生成。最大比例是$\frac{2^{112}-1}{1}=2^{112}-1$

2.3闪电贷

在Uniswap v1中，使用XYZ购买ABC的用户需要先将XYZ发送到合约，然后才能接收ABC。如果该用户需要他们要购买的ABC以获得他们要支付的XYZ，这将很不方便。例如，用户可能正在使用该ABC在某些其他合约中购买XYZ，以便从Uniswap套利价格差，或者他们可能通过出售抵押品以偿还Uniswap来平仓Maker或Compound的头寸。

Uniswap v2添加了一项新功能，即只要用户在同一原子交易中进行付款，用户就可以在付款之前接收和使用该资产。交换功能在转移出用户请求的代币和强制执行不变式之间调用可选的用户指定的回调协定。

如果合约没有足够的资金，它将还原整个交易。用户也可以使用相同的代币偿还Uniswap池，而不用完成交换。这实际上与允许任何人快速借入Uniswap池中存储的任何资产相同，收取与Uniswap交易费用相同的0.30%的费用。(5)

(5) 由于Uniswap对投入金额收取费用，因此相对于提款金额的费用实际上略高：$\frac{1}{1-0.003}-1=\frac{3}{997}\approx 0.3009203\%$。

2.4协议费用

Uniswap v2包含0.05%的协议费用，可以将其打开和关闭。如果启用，则此费用将发送到工厂合约中指定的FeeTo地址。

最初，未设置feeTo，并且不收取任何费用。预先指定的地址feeToSetter可以调用Uniswap v2工厂合约上的setFeeTo函数，将feeTo设置为其他值。feeToSetter也可以调用setFeeToSetter来更改feeToSetter地址本身。

如果设置了feeTo地址，则协议将开始收取5个基本点的费用，这是流动性提供商赚取的30个基本点的费用的$\frac{1}{6}$。也就是说，交易者将继续为所有交易支付0.30%的费用；该费用的83.3%（交易量的0.25%）将流向流动性提供者，而该费用的16.6%（交易量的0.05%）将流向该费用收件人。

在交易时收取这0.05%的费用会给每笔交易带来额外的gas成本。为避免这种情况，仅在存入或提取流动性时才收取累积费用。合约计算累计费用，并在铸造或销毁任何代币之前立即向受益者铸造新的流动性代币。

可以通过测量自上次收取费用以来的$\sqrt{k}$（即$\sqrt{x\cdot y}$）的增长来计算总费用(6)。公式为你提供了t_1和t_2之间累积的费用，以占其流动性的百分比表示。t_2的池：

$$f_{1,2} = 1 - \frac{\sqrt{k_1}}{\sqrt{k_2}}$$

如果费用是在t_1之前激活的，则feeTo地址应捕获在t_1和t_2之间累积的费用中的$\frac{1}{6}$。因此，我们想向新的流动性代币铸造到FeeTo地址，以表示池的$\phi\cdot f_{1,2}$，其中ϕ为$\frac{1}{6}$。

也就是说，我们要选择满足以下关系的s_m，其中s_1是时间t_1的流通股总数：

$$\frac{s_m}{s_m + s_1} = \phi\cdot f_{1,2}$$

经过一些操作，包括将$1 - \frac{\sqrt{k_1}}{\sqrt{k_2}}$替换为$f_{1,2}$并求解$s_m$，我们可以将其重写为：

$$s_m = \frac{\sqrt{k}_2 - \sqrt{k}_1}{(\frac{1}{\phi} - 1)\cdot \sqrt{k}_2 + \sqrt{k}_1}\cdot s_1$$

将ϕ设置为$\frac{1}{6}$用以下公式表达：

$$s_m = \frac{\sqrt{k}_2 - \sqrt{k}_1}{5\cdot \sqrt{k}_2 + \sqrt{k}_1}\cdot s_1$$

假设最初的存户将100个DAI和1个ETH放入一对，获得10股。一段时间后（没有其他任何存户参与该对），他们试图撤回该货币对，当时该对有96个DAI和1.5 ETH 。将这些值插入上面的公式可以得到以下结果

$$s_m = \frac{\sqrt{1.5\cdot 96} - \sqrt{1\cdot 100}}{5\cdot \sqrt{1.5\cdot 96} + \sqrt{1\cdot 100}}\cdot 10 \approx 0.0286$$

(6) 我们可以使用此不变式，它不考虑铸造或销毁的流动性代币，因为我们知道每次存入或提取流动性时都要收取费用。

2.5池份额的元交易

Uniswap v2对创建的池份额本机支持元交易。这意味着用户可以使用签名(7)，而不是通过其地址进行链上交易来授权其池份额的转移。任何人都可以通过调用许可功能，支付gas以及可能在同一笔交易中执行其他操作来代表用户提交此签名。

(7) 签名的消息符合EIP-712标准，与元交易令牌（如CHAI和DAI）使用的相同。

3.其他

3.1 Solidity

Uniswap v1是在Vyper，类似于Python的智能合约语言中实现的。Uniswap v2在更广泛使用的Solidity中实现，因为它需要Vyper中尚未提供的某些功能，例如，解释非标准ERC-20代币的返回值的功能，以及访问新操作码的功能 。

3.2合约重新架构

Uniswap v2的一个设计优先事项是最大程度地减少核心对合约（存储流动性提供者的资产的合约）的表面积和复杂性。该合约中的任何错误都可能是灾难性的，因为数百万美元的流动资金可能被盗或冻结。在评估此核心合约的安全性时，最重要的问题是它是否可以保护流动性提供者避免其资产被盗或锁定。可以用来支持或保护交易者的任何功能，允许将池中的一种资产交换为另一种的基本功能除外，都可以在"路由器"合约中处理。

实际上，甚至交换功能的一部分都可以拉到路由器合约中。如上所述，Uniswap v2存储每个资产的最后记录的余额，以防止对预言机机制进行特定的操纵性利用。新架构利用这一优势进一步简化了Uniswap v1合约。

在Uniswap v2中，卖方在调用掉期功能之前将资产发送到核心合约。然后，合约通过将最近记录的余额与当前余额进行比较，来衡量其已收到多少资产。

这意味着核心合约与交易者转移资产的方式无关。可以使用一个元交易或授权ERC-20转让的任何其他未来机制来代替transferFrom。

3.2.1 费用调整

Uniswap v1的交易费是通过在强制执行不变产品不变式之前将支付给合约的金额减少0.3%来收取的。合约隐含地执行以下公式：

$$(x_1 - 0.003 \cdot x_{in}) \cdot y_1 >= x_0 \cdot y_0$$

通过快速交换，Uniswap v2可能会导致x_{in}和y_{in}也许都可能不为零（当用户希望使用相同资产而不是交换来偿还货币对时）。为了在适当收费的情况下处理此类情况，签订合约以强制执行以下不变式：(8)

$$(x_1 - 0.003 \cdot x_{in}) \cdot (y_1 - 0.003 \cdot y_{in}) >= x_0 \cdot y_0$$

为了简化链上的计算，我们可以将不等式的每一边乘以1,000,000

$$(1000 \cdot x_1 - 3 \cdot x_{in}) \cdot (1000 \cdot y_1 - 3 \cdot y_{in}) >= 1000000 \cdot x_0 \cdot y_0$$

(8) 注意，使用新的体系结构，用户不提供x_{in}；相反，它是通过测量回调后的合约余额x_1并从中减去$\left(x_0 - x_{out}\right)$来计算的。这种逻辑不会区分在调用之前发送到合约中的资产和在回调期间发送到合约中的资产。y_{in}以相同的方式基于y_0,y_1,和y_{out}计算。

3.2.2 sync() 和 skim()

为了防止可能更新配对合约余额的定制令牌实现并更优雅地处理总供应量大于2^{112}的令牌，Uniswap v2具有两个纾困功能：sync()和skim()

如果令牌异步缩小一对货币对的余额，sync()充当恢复机制。在这种情况下，交易将获得次优利率，并且如果没有流动性提供者愿意纠正这种情况，则该交易对将被卡住。sync()存在，可以将合约的储备金设置为当前余额，从而可以从这种情况下略微恢复。

如果将足够的令牌发送到货币对以使两个uint112存储槽中的储备金溢出，skim()将用作恢复机制，否则可能导致交易失败.skim()允许用户提取货币对当前余额与货币对之间的差额，如果呼叫者的差值大于0，则为$2^{112} - 1$。

3.3处理非标准和异常代币ERC-20标准

要求transfer()和transferFrom()返回一个布尔值，指示调用成功或失败[4]。这些功能中的一个或两个都在某些代币（包括诸如Tether（USDT）和Binance Coin（BNB）等流行代币）上的实现却没有返回值。Uniswap v1将这些不正确定义的函数的缺失返回值解释为false（即，表示传输不成功的指示），并还原事务，从而导致尝试的传输失败。

Uniswap v2对非标准实现的处理方式有所不同。具体来说，如果transfer()call(9)没有返回值，则Uniswap v2会将其解释为成功而不是失败。此更改不应影响任何符合标准的ERC-20代币（因为在这些代币中，transfer()始终具有返回值）。Uniswap v1还假设对transfer()和transferFrom()的调用不能触发对Uniswap对合约的可重入调用。

某些ERC-20代币违反了这一假设，包括支持ERC-777的"钩子"[5]的代币。为了完全支持此类代币，Uniswap v2包含一个"锁"，可直接防止重新进入所有公共状态更改功能。

(9) 如上文第3.2节所述，Uniswap v2内核不使用transferFrom()。

3.4 初始化流动性代币供应

当新的流动性提供商将令牌存入现有的Uniswap对中时，将根据现有的令牌数量来计算铸造的流动性令牌的数量：

$$s_{minted} = \frac{x_{deposited}}{x_{starting}} \cdot s_{starting}$$

但是，如果他们是第一笔存款怎么办？ 在这种情况下，$x_{starting}$为0，因此该公式将不起作用。

Uniswap v1将初始股票供应量设置为等于所存入的以太坊（wei）的数量。这是一个合理的值，因为如果初始流动性是以正确的价格存入的，那么1个流动性池份额（与ETH类似，是18位精度代币）将价值约2 ETH。

但是，这意味着流动性池份额的价值取决于最初存入流动性的比率，这是相当任意的，特别是因为不能保证该比率反映了真实的价格。此外，Uniswap v2支持任意对，因此许多对根本不包含ETH。

相反，Uniswap v2最初铸造的份额等于所存金额的几何平均值：

$$s_{minted} = \sqrt{x_{deposited} \cdot y_{deposited}}$$

该公式可确保任何时候的流动资金池份额的值基本上与最初存入流动资金的比率无关。例如，假设1 ABC的价格当前为100 XYZ。如果初始存款为2 ABC和200 XYZ（比率为1：100），则存款人将获得$\sqrt{2 \cdot 200} = 20$股。这些股份现在仍应价值2 ABC和200 XYZ，外加累积费用。

如果初始存款为2 ABC和800 XYZ（比率为1：400），则存款人将收到$\sqrt{2 \cdot 800} = 40$份额(10)。

上面的公式可确保流动性池份额的价值永远不会低于该池中储备的几何平均值。但是，流动资金池份额的价值有可能随着时间的推移而增长，这可以通过累积交易费用或通过向流动资金池的“捐赠”来实现。从理论上讲，这可能导致最小数量的流动性池份额（1e-18池份额）的价值过高，以至于小型流动性提供者无法提供任何流动性。

为了缓解这种情况，Uniswap v2会刻录创建的第一个1e-15（0.000000000000001）池份额（池份额最小数量的1000倍），然后将它们发送到零地址，而不是发送到铸造者。对于几乎所有令牌对来说，这应该是微不足道的成本(11)。但是，这大大增加了上述攻击的成本。为了将流动资金池份额的价值提高到100美元，攻击者需要向该池捐赠100,000美元，该资金将永久锁定为流动资金。

(10) 这也减少了舍入误差的可能性，因为股份数量中的位数大约等于储备中资产X数量中位数的数量和储备中资产Y数量中位数的平均值。储备金：

$$\log_2 \sqrt{x \cdot y} = \frac{\log_2 x + \log_2 y}{2}$$

(11) 从理论上讲，这种销毁在某些情况下可能是不可忽略的，例如高值零小数令牌之间的配对。但是，无论如何，这些对都不适合Uniswap，因为舍入误差会使交易变得不可行。

3.5包装ETH

与以太坊的主资产ETH进行交易的界面不同于与ERC-20代币进行交互的标准界面。所以，以太坊上的许多其他协议不支持ETH，而是使用规范的"包装的ETH"代币WETH[6]。

Uniswap v1是一个例外。由于每对Uniswap v1对都将ETH作为一项资产，因此直接处理ETH是有意义的，这样可以节省更多的gas。

由于Uniswap v2支持任意ERC-20对，因此现在不再需要支持未封装的ETH。增加这样的支持将使核心代码库的大小增加一倍，并有可能导致ETH和WETH对之间流动性的分散化(12)。原始ETH需要包装在WETH中，然后才能在Uniswap v2上进行交易。

(12) 在撰写本文时，Uniswap v1上流动性最高的货币对之一是ETH和WETH之间的货币对[7]。

3.6确定性地址对

与Uniswap v1中一样，所有Uniswap v2对合约都由单个工厂合约实例化。在Uniswap v1中，这些对合约是使用CREATE操作码创建的，这意味着此类合约的地址取决于该对的创建顺序。Uniswap v2使用以太坊的新CREATE2操作码[8]生成具有确定性地址的成对合约。这意味着无需查看链的状态，就可以在链外计算对的地址。

3.7最大代币余额

为了有效地实现预言机机制，Uniswap v2仅支持高达$2^{112} - 1$的备用余额。此数字足以支持18个小数位数的代币，其总供应量超过1千万。如果任何一个准备金余额确实超过2^{112}，则对swap函数的任何调用将开始失败。为了从这种情况中恢复过来，任何用户都可以调用skim()函数从流动资金池中删除多余的资产。

参考资料

[1]Hayden Adams. 2018.https://hackmd.io/@477aQ9OrQTCbVR3fq1Qzxg/HJ9jLsfTz?type=view.
[2]Guillermo Angeris et al. An analysis of Uniswap markets. 2019. arXiv:[1911.03380 q-fin. TR] https://arxiv.org/abs/1911.03380.
[3]samczsun.Taking undercollateralized loans for fun and for profit. Sept.2019.https://samczsun.com/taking-undercollateralized-loans-for-fun-and-for-profit.
[4]Fabian Vogelsteller and Vitalik Buterin. Nov. 2015.https://eips.ethereum.org/EIPS/eip-20.
[5]Jordi Baylina Jacques Dafflon and Thomas Shababi.EIP 777: ERC777 Token Standard.Nov. 2017.https://eips.ethereum.org/EIPS/eip-777.
[6]Radar.WTF is WETH?https://weth.io.
[7]Uniswap.info.Wrapped Ether (WETH).https://uniswap.info/token/0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2.
[8]Vitalik Buterin.EIP 1014: Skinny CREATE2. Apr. 2018.https://eips.ethereum.org/EIPS/eip-1014.

4 免责声明

本文仅供参考。它不构成买卖任何投资的投资建议或建议或招揽，也不应用于评估作出任何投资决定的优劣。不应将其作为会计，法律或税务建议或投资建议的依据。本文仅反映作者当前的观点，并不代表Paradigm或其关联公司，也不一定反映Paradigm，其附属公司或与Paradigm相关的个人的观点。本文中反映的观点如有更改，恕不更新。