Uniswap v2 核心

海登·亚当斯
hayden@uniswap.org

Noah Zinsmeister
noah@uniswap.org

丹·罗宾逊
dan@paradigm.xyz

2020 年 3 月

抽象的

本技术白皮书解释了 Uniswap v2 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 ERC20 之间的任意配对、允许其他合约估算给定时间间隔内时间加权平均价格的强化价格预言机、允许交易者接收资产并在其他地方使用,然后在交易后期支付的“闪电交换”,以及可以在未来启用的协议费用。它还重新设计了合约以减少其攻击面。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 是以太坊区块链上的链上智能合约系统,基于“恒定乘积公式” [1] 实现自动化流动性协议。每个 Uniswap v1 对存储两种资产的集合储备,并为这两种资产提供流动性,保持储备乘积不会减少的不变量。交易者在交易中支付 30 个基点的费用,这些费用将支付给流动性提供者。合约不可升级。

Uniswap v2 是基于相同公式的新实现,具有几个非常受欢迎的新功能。最重要的是,它允许创建任意的 ERC20/ERC20 对,而不是仅支持 ERC20 和 ETH 之间的对。它还提供了一个强化的价格预言机,在每个区块开始时累积两种资产的相对价格。这允许以太坊上的其他合约在任意时间间隔内估算两种资产的时间加权平均价格。最后,它支持“闪电交换”,用户可以自由接收资产并在链上的其他地方使用它们,只需在交易结束时支付(或退还)这些资产。

虽然合约通常不可升级,但有一个私钥可以更新工厂合约上的变量,以开启链上 5 个基点的交易费用。此费用最初将被关闭,但未来可能会开启,此后流动性提供者将在每笔交易中赚取 25 个基点,而不是 30 个基点。

如第 3 节所述,Uniswap v2 还修复了 Uniswap v1 中的一些小问题,并重新设计了实现,减少了 Uniswap 的攻击面,并通过最小化持有流动性提供者资金的“核心”合约中的逻辑使系统更容易升级。

本文介绍了该核心合约的机制,以及用于实例化这些合约的工厂合约。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 作为桥梁货币。每对交易都包含 ETH 作为其资产之一。这使得路由更简单 ABC 和 XYZ 之间的每笔交易都经过 ETH/ABC 对和 ETH/XYZ 对 并减少了流动性的分散。

然而,这条规则给流动性提供者带来了巨大的成本。所有流动性提供者都持有 ETH,并会因其他资产相对于 ETH 的价格变化而遭受无常损失。当两种资产 ABC 和 XYZ 相互关联时（例如,如果它们都是美元稳定币）,Uniswap 对 ABC/XYZ 上的流动性提供者通常比 ABC/ETH 或 XYZ/ETH 对遭受的无常损失要少。

使用 ETH 作为强制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 ABC/XYZ 对的两倍,并且会遭受两次滑点。

Uniswap v2 允许流动性提供者为任意两个 ERC-20 创建配对合约。
任意 ERC-20 之间的货币对激增可能会使找到交易特定货币对的最佳路径变得更加困难,但可以在更高层（链下或通过链上路由器或聚合器)处理路由。

2.2 价格预言机

可以计算出 Uniswap 在时间 t 提供的边际价格（不包括费用）
将资产 a 的储备除以资产 b 的储备。

$$\text{点} = \frac{r_A}{tb\ r\ t} \tag{1}$$

由于如果价格不正确（误差足以弥补费用）,套利者将与 Uniswap 进行交易,因此 Uniswap 提供的价格往往会跟踪资产的相对市场价格,正如 Angeris 等人所表明的那样 [2]。这意味着它可以用作近似价格预言机。

但是,Uniswap v1 并不适合用作链上价格预言机,因为它很容易被操纵。假设其他合约使用当前的 ETH-DAI 价格来结算衍生品。想要操纵测量价格的攻击者可以从 ETH-DAI 对中购买 ETH,触发衍生品合约的结算（使其根据虚高的价格结算）,然后将 ETH 卖回给该对以将其交易回真实价格。1 这甚至可以作为原子交易完成,或者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区块第一笔交易之前的价格（或者,在前一个区块的最后一笔交易之后)改进了这一预言机功能。

1 有关使用 Uniswap v1 作为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]。

区块价格（即区块内的价格)比区块内的价格更难操纵。如果攻击者提交交易试图操纵区块末尾的价格,其他套利者可能能够在同一区块中立即提交另一笔交易进行交易。矿工（或使用足够多 gas 填充整个区块的攻击者)可以在区块末尾操纵价格,但除非他们也开采下一个区块,否则他们可能在套利交易方面没有特别的优势。

具体来说,Uniswap v2 通过跟踪每个区块开始时某人与合约交互的价格累计总和来累计此价格。根据区块时间戳,每个价格都按自上次更新该价格的区块以来经过的时间加权。2 这意味着任何给定时间（更新后)的累加器值应为合约历史上每秒的现货价格总和。

$$\text{在} = \prod_{\text{我}=1}^{\text{吨}} \pi \tag{2}$$

为了估算从时间t1到t2 的时间加权平均价格,外部调用者可以在t1处检查累加器的值,然后在t2 处再次检查,从第二个值中减去第一个值,然后除以经过的秒数。（请注意,合约本身不存储此累加器的历史值 - 调用者必须在期间开始时调用合约来读取和存储此值。）

$$pt1,t2 = \frac{\text{t2}}{\text{i = t1}\pi\text{t2}} = \frac{\text{t2}}{\text{i=1 pi} - \text{at2} - \text{at1 t1 i=1}\pi\text{t1预言机用户可以选择}} \tag{3}$$

$$\text{— t1}$$

何时开始和结束该周期。选择较长的周期会使攻击

者操纵 TWAP 的成本更高,尽管这会导致价格不太及时。

一个复杂因素是:我们应该用资产 B 来衡量资产 A 的价格,还是用资产 A 来衡量资产 B 的价格?虽然 A 相对于 B 的现货价格始终是 B 相对于 A 的现货价格的倒数,但在特定时期内,资产 A 相对于资产 B 的平均价格并不等于资产 B 相对于资产 A 的平均价格的倒数。3例如,如果 USD/ETH 价格在区块 1 中为 100,在区块 2 中为 300,则 USD/ETH 的平均价格将为 200 USD/ETH,但 ETH/USD 的平均价格将为 1/150 ETH/USD。由于合约无法知道用户希望使用哪一种资产作为记账单位,因此 Uniswap v2 会跟踪这两种价格。

另一个复杂因素是,有人可能会将资产发送到配对合约,从而更改其余额和边际价格,而无需与其交互,因此不会触发预言机更新。如果合约只是检查自己的余额并根据当前价格更新预言机,攻击者可以通过在区块中第一次调用合约之前立即将资产发送到合约来操纵预言机。如果最后一笔交易发生在时间戳为 X 秒前的区块中,合约会在累积新价格之前错误地将新价格乘以 X,即使没有人

2由于矿工可以自由设置区块时间戳,因此预言机用户应该注意
这些值可能与真实世界的时间不完全对应。

3 资产 A 在给定时间段内相对于资产 B 的算术平均价格等于资产 B 在给定时间段内相对于资产 A 的调和平均价格的倒数。如果合约衡量的是几何平均价格,那么这两个价格就是彼此的倒数。然而,几何平均 TWAP 不太常用,而且在以太坊上很难计算。

有机会以该价格进行交易。为了防止这种情况发生,核心合约在每次交互后都会缓存其储备,并使用从缓存储备而不是当前储备得出的价格更新预言机。除了保护预言机免受操纵之外,这一变化还支持下面第 3.2 节中描述的合约重新架构。

2.2.1 精度

由于 Solidity 对非整数数字数据类型没有一流的支持,因此 Uniswap v2 使用简单的二进制定点格式来编码和操纵价格。

具体来说,特定时刻的价格存储为 UQ112.112 数字,这意味着小数点两边指定 112 个小数位精度,没有符号。

这些数字的范围为 [0, 2112 − 1]4 ,精度为 UQ112.112 格式是出于务实的原因而选择的 因 $\overline{1\,2\,11\,2}$。

为这些数字可以存储在 uint224 中,这样 256 位存储槽中就有 32 位是空闲的。另外,每个存储在 uint112 中的保留位也会在（打包的)256 位存储槽中留下 32 位空闲位。

这些可用空间用于上述累积过程。具体来说,储备与最近至少有一笔交易的区块的时间戳一起存储,用232取模,使其适合 32 位。此外,虽然保证任何给定时刻的价格（存储为 UQ112.112 数字)适合 224 位,但该价格在一段时间内的累积却不能。存储槽末尾的额外 32 位用于存储因重复求和价格而产生的溢出位。这种设计意味着价格预言机只会在每个区块的第一笔交易中添加额外的三个 SSTORE 操作（当前成本约为 15,000 gas）。

主要缺点是 32 位不足以存储永远不会溢出的时间戳值。事实上,Unix 时间戳溢出 uint32 的日期是 02/07/2106。为了确保此系统在此日期之后以及此后每232 − 1 秒的倍数继续正常运行,预言机只需在每个间隔（大约 136 年)至少检查一次价格即可。这是因为累积的核心方法（和时间戳的修改)实际上是溢出安全的,这意味着只要预言机使用正确的（简单)溢出算法来计算增量,就可以适当地考虑溢出间隔内的交易。

2.3 闪电互换

在 Uniswap v1 中,使用 XYZ 购买 ABC 的用户需要先将 XYZ 发送到合约,然后才能收到 ABC。如果该用户需要购买 ABC 才能获得用于支付的 XYZ,那么这将非常不方便。例如,用户可能使用该 ABC 在其他合约中购买 XYZ,以从 Uniswap 套利差价,或者他们可以通过出售抵押品来偿还 Uniswap,从而平仓 Maker 或 Compound 的头寸。

Uniswap v2 增加了一项新功能,允许用户在支付之前接收和使用资产,只要他们在同一原子交易中进行支付即可。交换函数在转出用户请求的代币和执行不变量之间调用可选的用户指定回调合约。回调完成后,合约会检查新余额并确认满足不变量

4 理论上限为2112 − ($\overline{1\,2112}$) 不适用于此设置,因为 UQ112.112 在2 112−1 1中数字 Uniswap 总是由两个 uint112 的比率生成。最大的比率是 $\frac{}{}$ = 2112 − 1。

（根据支付金额调整费用后）。如果合约资金不足，则会撤销整个交易。

用户还可以使用相同的代币偿还 Uniswap 池，而不是完成交换。这实际上等同于让任何人以闪电借贷方式借入存储在 Uniswap 池中的任何资产（费用与 Uniswap 收取的交易费用相同，为 0.30%）。5

2.4 协议费用

Uniswap v2 包含 0.05% 的协议费，可以开启或关闭。如果开启，
该费用将被发送到工厂合同中指定的 feeTo 地址。

最初，feeTo 未设置，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 设置为不同的值。feeToSetter 也可以调用 setFeeToSetter 来更改 feeToSetter 地址本身。

如果设置了 feeTo 地址，协议将开始收取 5 个基点的费用，这是流动性提供者赚取的 30 个基点费用的提成。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%（交易金额的 0.25%）将支付给流动性提供者，而该费用的 16.6%（交易金额的 0.05%）将支付给 feeTo 地址。

在交易时收取这 0.05% 的费用将对每笔交易产生额外的 gas 成本。为了避免这种情况，仅在存入或提取流动性时才收取累积费用。合约计算累积费用，并在铸造或销毁任何代币之前立即向费用受益人铸造新的流动性代币。

可以通过测量自上次收取费用以来 $\sqrt{k}$（即 $\sqrt{x \cdot y}$）的增长来计算所收取的总费用。6此公式为您提供t1和t2之间的累积费用占t2时池中流动性的百分比：

$$f_{1,2} = 1 - \sqrt{\frac{k_2}{k_1}} \tag{4}$$

如果费用在t1 之前激活，则 feeTo 地址应捕获 $\frac{1}{6}$ 在t1和t2之间累积的费用。
因此，我们希望向 feeTo 地址铸造新的流动性代币，代表池子的 $\phi \cdot f_{1,2}$，其中 ϕ 是 $\frac{1}{6}$。

也就是说，我们希望选择sm来满足以下关系，其中s1是总
t1时刻流通股数：

$$\frac{\text{山猫}}{\text{+ s1}} = \phi \cdot f_{1.2}\ sm \tag{5}$$

经过一些操作，包括用 $1 - \sqrt{k_2}$ 代入 $\sqrt{\frac{k_2}{k_1}}$ 对于f1,2并求解sm，我们
可以将其重写为：

$$\text{山姆} = \frac{\sqrt{k_2} - \sqrt{\frac{k_2}{k_1}}}{\frac{1}{\phi}\left(k_1 - 1\right) \cdot \sqrt{k_2} + \sqrt{k_1}} \cdot s_1 \tag{6}$$

将 ϕ 设置为 $\frac{1}{6}$ 给出以下公式：

5由于 Uniswap 对输入金额收取费用，因此相对于提取金额的费用实际上略高：$\frac{1}{1 - 0.003} - 1 = \frac{3}{997} \approx 0.3009203\%$。

6我们可以利用这个不变量，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会收取费用。

$$山姆= \frac{\sqrt{k2} - \sqrt{\quad}}{k1\,5 \cdot \sqrt{\quad} k2 + \sqrt{\quad} k1} \cdot s1 \tag{7}$$

假设初始存款者将 100 DAI 和 1 ETH 放入一对,获得 10 股。
过了一段时间（没有其他存款人参与该对）,他们试图提取这笔钱,此时该对有 96 DAI 和 1.5 ETH。将这些值代入上述公式,我们得到以下结果：

$$山姆= \frac{\sqrt{1.5 \cdot 96} - \sqrt{1 \cdot 100 \cdot}}{1.5 \cdot 96 + \sqrt{1 \cdot 100}} \approx 0.0286\,5 \cdot \sqrt{\quad} \tag{8}$$

2.5 池份额的元交易

Uniswap v2 对铸造的池份额原生支持元交易。这意味着用户可以使用签名7 授权转移池份额,而不是从其地址进行链上交易。任何人都可以通过调用 permit 函数、支付 gas 费用并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。

3 其他变化

3.1 坚固性

Uniswap v1 是用 Vyper（一种类似 Python 的智能合约语言)实现的。Uniswap v2 是用更广泛使用的 Solidity 实现的,因为它需要一些 Vyper 在开发时尚未提供的功能（例如,能够解释非标准 ERC-20 代币的返回值,以及通过内联汇编访问新的操作码,例如 chainid）。

3.2 合约重构

Uniswap v2 的一个设计重点是尽量减少核心对合约（存储流动性提供者资产的合约)的表面积和复杂性。此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性可能会被盗或冻结。

在评估此核心合约的安全性时,最重要的问题是它是否能保护流动性提供者的资产免遭盗窃或锁定。除了允许将池中的一种资产兑换为另一种资产的基本功能之外,任何旨在支持或保护交易者的功能都可以在“路由器”合约中处理。

事实上,甚至部分交换功能也可以被拉入路由器合约中。
如上所述,Uniswap v2 存储了每项资产的最后记录余额（以防止对预言机机制的特定操纵利用）。新架构利用这一点进一步简化了 Uniswap v1 合约。

在 Uniswap v2 中,卖方在调用交换函数之前将资产发送到核心合约。然后,合约通过比较最后记录的余额与当前余额来衡量已收到的资产数量。这意味着核心合约是不可知的

7签名消息符合 EIP-712 标准,与元交易使用的标准相同
像 CHAI 和 DAI 这样的代币。

交易者转移资产的方式。除了 transferFrom 之外,它还可以是元交易,或任何其他未来授权转移 ERC-20 的机制。

3.2.1 费用调整

Uniswap v1 的交易费是通过在执行常数乘积不变量之前将支付给合约的金额减少 0.3% 来应用的。合约隐式执行以下公式:

$$(x1 - 0.003 \cdot xin)) \cdot y1 >= x0 \cdot y0 \quad (9)$$

通过闪电交换,Uniswap v2 引入了xin和yin可能都为非零的可能性（当用户希望使用相同资产而不是交换来偿还该对时）。为了在正确应用费用的同时处理此类情况,合约被编写为强制执行以下不变量:8

$$(x1 - 0.003 \cdot xin) \cdot (y1 - 0.003 \cdot yin) >= x0 \cdot y0 \quad (10)$$

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:

$$(1000 \cdot x1 - 3 \cdot xin) \cdot (1000 \cdot y1 - 3 \cdot yin) >= 1000000 \cdot x0 \cdot y0 \quad (11)$$

3.2.2 sync() 和 skim()

为了防止定制代币实现更新配对合约的余额,并更优雅地处理总供应量可能大于2112 的代币,Uniswap v2 有两个救助函数:sync() 和 skim()。 ,

如果代币异步减少某对的余额,sync() 可充当恢复机制。在这种情况下,交易将获得次优利率,如果没有流动性提供者愿意纠正这种情况,该对就会被卡住。sync() 存在用于将合约的储备设置为当前余额,从而从这种情况中提供某种优雅的恢复。

skim() 可充当恢复机制,以防向某个货币对发送了足够多的代币,从而溢出两个 uint112 存储槽的储备,否则可能导致交易失败。如果差额大于 0， skim() 允许用户将该货币对的当前余额与2112 - 1 之间的差额提取给调用者。

3.3 处理非标准和不寻常的标记

ERC-20 标准要求 transfer() 和 transferFrom() 返回一个布尔值,指示调用是成功还是失败 [4]。但某些代币（包括 Tether (USDT) 和 Binance Coin (BNB) 等热门代币)上的这两个函数中的一个或两个的实现却没有返回值。Uniswap v1 将这些定义不当的函数的缺失返回值解释为 false（即表示转账不成功）,并还原交易,导致尝试的转账失败。

8请注意,使用新架构时， xin不是由用户提供的;而是通过测量回调后的合约余额x1并从中减去 (x0 - xout)来计算的。此逻辑不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。yin以相同的方式计算,基于y0、 y1和yout。

Uniswap v2 对非标准实现的处理方式不同。具体来说,如果 transfer() 调用[9]没有返回值,Uniswap v2 会将其解释为成功而不是失败。此更改不应影响符合标准的任何 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”的代币 [5]。为了完全支持此类代币,Uniswap v2 包含一个“锁”,可直接防止所有公共状态更改函数的重入。这还可以防止从 flash 交换中的用户指定的回调进行重入,如第 2.3 节所述。

3.4 流动性代币供应初始化

当新的流动性提供者将代币存入现有的 Uniswap 对时,数量流动性代币的铸造量是根据现有代币数量计算的：

$$\text{已铸成} = \frac{\text{xdeposited} \cdot}{\text{开始}}\text{开始} \tag{12}$$

但如果他们是第一批存款人呢?在这种情况下， xstarting为 0,因此此公式将不起作用。

Uniswap v1 将初始份额供应量设置为等于存入的 ETH 数量（以 wei 为单位）。这是一个相当合理的值,因为如果初始流动性以正确的价格存入,那么 1 个流动性池份额（与 ETH 一样,是 18 进制代币)的价值约为 2 ETH。

然而,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,这相当随意,尤其是因为无法保证该比率反映真实价格。此外,Uniswap v2 支持任意对,因此许多对根本不包含 ETH。

相反,Uniswap v2 最初铸造的股份等于存入金额的几何平均值：

$$\text{sminted} = \sqrt{\text{xdeposited} \cdot \text{ydeposited}} \tag{13}$$

此公式确保流动性池份额的价值在任何时候都基本独立于流动性最初存入的比率。例如,假设 1 ABC 的价格目前为 100 XYZ。如果初始存款为 2 ABC 和 200 XYZ（比率为 1:100）,存款人将收到 $\sqrt{2 \cdot 200} = 20$ 股。这些股份现在的价值仍应为 2 ABC 和 200 XYZ,加上累计费用。

如果初始存款为 2 ABC 和 800 XYZ（比例为 1:400）,存款人将获得 $\sqrt{2 \cdot 800} = 40$ 池份额。[10]上述公式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。然而，

[9]如上文3.2节所述,Uniswap v2核心不使用transferFrom()。

[10]这也降低了舍入误差的可能性,因为股票数量的位数大约等于储备中资产 X 数量的位数与储备中资产 Y 数量的位数的平均值：

$$\log_2 \sqrt{x \cdot y} = \frac{\log_2 x + \log_2 y}{2} \tag{14}$$

流动性池份额会随着时间的推移而增长,要么通过累积交易费,要么通过向流动性池“捐赠”。理论上,这可能会导致这样一种情况:最低数量的流动性池份额(1e-18 池份额)的价值如此之高,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 会销毁铸造的前 1e-15(0.000000000000000001)个池份额(池份额最小数量的 1000 倍),并将其发送到零地址而不是铸造者。对于几乎任何代币对来说,这应该是可以忽略不计的成本。11但它大大增加了上述攻击的成本。为了将流动性池份额的价值提高到 100 美元,攻击者需要向池中捐赠 100,000 美元,这些资金将被永久锁定为流动性。

3.5 包装 ETH

与以太坊原生资产 ETH 进行交易的接口与与 ERC-20 代币交互的标准接口不同。因此,以太坊上的许多其他协议都不支持 ETH,而是使用标准的“包装 ETH”代币 WETH [6]。

Uniswap v1 是个例外。由于每个 Uniswap v1 对都包含 ETH 作为一项资产,因此直接处理 ETH 是有意义的,这样可以稍微节省 gas。

由于 Uniswap v2 支持任意 ERC-20 对,因此现在不再需要支持未包装的 ETH。添加此类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 对之间的流动性分散12。原生 ETH 需要先包装成 WETH,然后才能在 Uniswap v2 上交易。

3.6 确定性对地址

与 Uniswap v1 一样,所有 Uniswap v2 配对合约都由单个工厂合约实例化。在 Uniswap v1 中,这些配对合约是使用 CREATE 操作码创建的,这意味着此类合约的地址取决于该配对的创建顺序。Uniswap v2 使用以太坊的新 CREATE2 操作码 [8] 来生成具有确定性地址的配对合约。这意味着可以在链下计算配对的地址(如果存在),而无需查看链状态。

3.7 最大代币余额

为了有效实现预言机机制,Uniswap v2 仅支持最高2112 − 1 的储备余额。这个数字足够高,可以支持总供应量超过 1 千万亿的 18 位小数代币。

如果任一储备余额确实超过2112 − 1,则对 swap 函数的任何调用都将开始失败(由于 _update() 函数中的检查)。要从这种情况中恢复,任何用户都可以调用 skim() 函数从流动资金池中移除多余的资产。

11理论上,在某些情况下,这种销毁是不可忽略的,例如高价值零小数代币之间的配对。然而,这些配对无论如何都不适合 Uniswap,因为舍入误差会使交易变得不可行。

12截至撰写本文时,Uniswap v1 上流动性最高的交易对之一是 ETH 和 WETH 之间的交易对 [7]。

参考

[1] Hayden Adams。2018 年。网址:https: //hackmd.io/@477aQ9OrQTCbVR3fq1Qzxg/HJ9jLsfTz?类型=视图。

[2] Guillermo Angeris 等人。Uniswap 市场分析。2019 年。arXiv: 1911.03380 [q-fin.TR].

[3] samczsun。以低额抵押贷款为乐,以盈利为目的。2019 年 9 月。网址： https://samczsun.com/taking-undercollateralized-loans-for-fun-and- for-profit /。

[4] 法比安·沃格尔斯特勒和维塔利克·布特林。 2015 年 11 月。网址： https://eips.ethereum。 org/EIPS/eip-20.

[5] Jordi Baylina Jacques Dafflon 和 Thomas Shababi。EIP 777:ERC777 代币标准。十一月2017。网址： https://eips.ethereum.org/EIPS/eip-777。

[6] 雷达。WETH 是什么?网址： https://weth.io/。

[7] Uniswap.info。Wrapped Ether（WETH）。网址:https: //uniswap.info/token/0xc02aaa39b223fe8d0a0e5c4f27 [8] Vitalik Buterin。EIP 1014:Skinny CREATE2。2018 年 4 月。网址： https://eips.ethereum。org/EIPS/eip-1014。

4 免责声明

本文仅供一般参考。它不构成投资建议或购买或出售任何投资的推荐或邀请,也不应用于评估任何投资决策的优点。它不应被依赖为会计、法律或税务建议或投资建议。本文反映了作者的当前观点,不代表 Paradigm 或其附属公司的观点,也不一定反映 Paradigm、其附属公司或与 Paradigm 相关的个人的观点。本文所反映的观点可能会发生变化,但不会更新。