从4337到7702：深入解读以太坊账号抽象赛道的过去与未来

前言

本文整体分两大模块：

上半段，将从2015年起的首个AA提案出发，系统整理目前为止的Eip提案主要内容，期望由史出发挖掘AA历史提案的历程，并综合性评价各方案优缺点。

下半段，着重对比EIP4337提出之后面临的市场低迷反馈，再深入分析如今即将被纳入下个版本以太坊升级的EIP7702，此提案一旦合并，将全方位改变链上应用形态。

EIP-7702 具有划时代的改变，且听十四君细细讲来

## 1、账号抽象的背景

### 1.1 账号抽象的意义定位

以太坊创始人vitalik在2023年年底再次更新 ETH 发展路线图，但其中针对账号抽象的设定，并未改过。如今的主流模式也正是从EIP-4337，步入到下一个阶段VoluntaryEOA Conversion（自愿转换EOA账号）。



https://x.com/VitalikButerin/status/1741190491578810445

在EIP4337推出一年多以来（在2023.3.1号丹佛的 WalletCon 上，官宣由以太坊基金会开发人员设计实现的ERC-4337 的核心合约已经通过了 OpenZeppelin 的审计，被认为是正式推出的历史节点）。

始终是只得到用户的广泛认可，但并不被广泛使用，如此矛盾的市场环境下，让EIP-7702的进度被大幅提前，乃至已经被确认将在下一次升级被合并其中。

### 1.2 账号抽象的市场现状

无需多言，直接看数据吧。

经过一年半的发展，EIP4337在主流链账号的集合下，仅仅有1200W的地址数，其中最为让人惊讶的是在以太坊主网上，活跃地址仅仅6,764个，或许统计维度有所问题，但至少与EOA与CA的地址数相差甚广，要知道以太坊主网上独立地址数已经达到2亿7千万（数据源于：https://etherscan.io/chart/address）。

可以说在主网上EIP4337是毫无实质性发展。



（图表数据来源：https://dune.com/niftytable/account-abstraction）

不过，这并不磨灭AA的本质价值，因为他从一开始的EIP4337的设计就注定了，他面对主网严重的往前兼容性问题上无法做好，所以伴随着各类L2层链的普遍嵌入原生AA，EIP4337的地址数在L2上获得爆发，其中base和polygon链的7月月度活跃用户分别是100W和300W，倒也颇为可观。

所以，并不是EIP4337设计错了，他有很多优点，我们一会会系统的总结，如今的现状是源于主网与L2之间的差异，他们需要用各自适合的方案。

## 2、账号抽象是什么？

账户抽象，听着很费解，但其实本质解决的是产权分离的问题。

EVM架构（即以太坊虚拟机）中有两种账户，外部账户（EOA）和合约账户（Contract Account），外部账户的所有权和签名权其实上是同一个体单位持有的。持有私钥的人不只拥有这个账户的「所有权」，同时还有权利「签名转移所有资产」。

这是由以太坊账号交易结构决定的

从下图的结构中可以发现，其实以太坊的标准交易是没有From方的，那么我做了一次资金转账，具体消费的是什么地址上的资金？实际是是通过其VRS参数（即用户签名）反解析出From地址的。

这里涉及到ECDSA等非对称加密，单向门限函数等概念，咱们不做展开，总之这里是由密码学来保障安全性，当然这也就造成了如今的产权合并的EOA地址窘境。

而EIP4337的核心效果，就是在交易字段里增加了Sender Address字段，从而能让私钥与被操作的地址分离开。



那为什么产权分离这么重要呢？

因为外部账户（EOA）设计会衍伸出更多的问题：

私钥难保护：用户失去私钥（遗失、黑客攻击、密码学上的被破解）意味着地失去所有资产。

签名算法少：原生协议在验证交易上只能使用 ECDSA 签名和验签算法。

签名权限高：无原生多签（多签只能通过智能合约实现协作），单签即可执行任意操作。

交易手续费只能通过 ETH 支付，并不支持批量交易。

交易隐私泄露：一对一交易容易分析账户持有者的隐私信息。

上诉的约束让普通用户很难使用以太坊：

首先，使用以太坊上的任何应用，用户都必须持有以太（并承担以太价格波动的风险）。

其次，用户需要处理复杂的费用逻辑，Gas price、Gas limit、事务阻塞（Nonce顺序）这些概念对用户来说过于复杂。

最后，虽然许多区块链钱包或应用试图通过产品优化提高用户体验，但它们的实际效果甚微。

所以破局之道在于实现账户抽象，将所有权（Owner）和签名权（Signer）解耦（Decoupling），从而才能逐个解决上述问题。

其实历史的方案有很多，最终都会汇聚到两种路线

## 3、AA历史提案脉络梳理



问题的解法看似有很多的EIP提案，但归根究底，就是两种核心思路，所以过往每一个没有被通过的EIP，其考虑的问题也就汇聚成了现在方案的破局之道。

### 3.1 第一种路线是让EOA地址变为CA地址

早在2015年11月15日，围绕EIP-101，Vitalik 就提出以合约作为账户的新结构。将地址改为只有代码和存储空间，改变手续费支持由ERC20支付，通过预编译合约将原生代币改为类ERC20来存余额（可具备代扣授权等功能），将交易字段精简到只有to、startgas、data和code。

现在看来，简直是大跃进式变革，会大幅改动底层设计，让每个账户地址都拥有自己的“代码”逻辑（其实也正是现在EIP-7702要做到的效果）。

还能衍生出其他的功能，比如

让交易使用更多加密算法，可以由各地址内部Code来指定验签鉴权方法

具备抗量子攻击特性，因为代码具备升级特性。

让以太币具备与ERC20合约一致的功能特性，核心效果有了代扣授权，从而无需原生币的损耗

提升账户的自定义空间，兼容社交恢复、sbt支持、密钥找回等

没有继续推进的原因也很简单，显然是步伐太大，对于当前交易哈希冲突问题，安全性隐患考虑不周所以一直搁置，但每个优点的理念都成为后续EIP4337以及EIP7702的核心功能之一。

后来还有一系列EIP试图完善这种逻辑

EIP-859：主链账户抽象--2018-01-30

试图解决Code的部署问题，核心作用是，如果出现了若交易方合约未部署，则使用交易附带code参数执行合约钱包部署，其次还提出新的 PAYGAS 操作码，除了支付gas外，也成为一笔交易的参数中验证部分与执行部分的分隔符。

虽然当时无疾而终，但是这也成了现在EIP7702的核心逻辑之一，EIP7702的每一笔交易结合特殊的交易结构，可以附带一定的代码，从而在本次交易中让EOA地址拥有合约能力。

EIP-7702：设置 EOA 账户代码 2024-05-07

这也是本文后续讨论机制的核心EIP，由Vitalik 发表了 EIP-7702 作为 EIP-3074 的替代方案（2024-05-07）。因此EIP-3074 被弃用，EIP-7702 被确定在即将到来的 ETH Prague/Electra（Pectra）硬分叉中纳入，具体内容咱们在下文展开。

### 3.2 第二种路线是让EOA地址驱动CA地址

EIP-3074：增加AUTH 和AUTHCALL 操作码--2020-10-15

在EVM 中加入两个新的OpCodesAUTH和AUTHCALL，让EOA 能透过这两个opcode 授权合约代替EOA 的身份去呼叫其他合约。

结合下图，概括来说一个EOA 能将一则已签的消息（交易）送至自己信任的合约（称作Invoker）上，此Invoker合约可以利用AUTH和AUTHCALL操作码来代替这个EOA 送出这笔交易。

EIP-4337：用交易内存池实现账户抽象--2021-09-29

总之，他受到MEV的启发进行设计，其核心价值是可以完全避免共识层协议更改。

eip4337提出新的事务对象UserOperation，用户将此对象发送到内存池中，由bundlers从矿工维度批量打包交付合约执行交易事务，本质上是把底层的交易与帐户运作拉到合约层面执行。

EIP-5189：通过背书人来操作抽象账户—2022-06-29

这算是优化了EIP4337的逻辑，是面对恶意的Bundler 通过建立资金罚款背书endorser的机制来防止Dos阻塞攻击。

### 3.3 其他用于支持AA的提案

EIP-2718：新交易类型的包装信封--2020-06-13

这倒是一个已经Final的提案，他定义一个新的交易类型，作为未来新增的交易类型的信封。

最终效果是，当引入新的交易类型时， 通过特定编码来区分这是哪一种交易，让其只需有向后兼容性，而无需往前兼容。最常见的例子就是EIP1559了，他区分了交易的手续费，使用了新的交易类型编码，又不影响最初的legacy的交易类型。

EIP-3607：让EOA地址不可部署合约--2021-06-10

这是是AA路径上的补充方案，用于防止合约部署地址与EOA地址冲突的问题。他会控制合约生成方法，让系统不允许将代码部署到已经是 EOA 地址的地址上。这个风险其实很小，毕竟以太坊地址有 160 位长，虽然存在用私钥碰撞出指定合约地址私钥的方法，但以比特币全算力投入估计，也还需要一年的时间。

### 3.4 如何理解账号抽象发展历程？

首先需要理解转为CA后的价值

基本上也就是EIP-4337的实际效果，他可以实现



但是，EIP-4337的核心缺点是违背人性动机原则。

他看起来是更好了，但是陷入了一种市场发展的死循环，Dapp很多还不兼容，那用户就不乐意使用CA地址，甚至使用CA有更高的交易成本（普通转账场景，也会交易费用翻倍），也太依赖于Dapp本身的兼容性。

所以在以太坊主网上迄今为止始终没有得到普及。

成本就是用户最重要衡量的标准，必须降低成本。

但是要真正降低GAS，就必须以太坊本身做软分叉升级，修改GAS计算或者修改操作码的GAS消耗等模块，然而既然要软分叉，那何不直接考虑EIP-7702呢？

## 4、全面解析EIP-7702

### 4.1 EIP-7702是什么

它通过新的交易类型来区分，可以允许EOA在单笔交易中临时具备智能合约的功能，进而支持业务上进行批量交易、无Gas交易和自定义权限管理等，且无需引入新的EVM opCode（影响往前兼容性）。

他可以让用户在不部署智能合约的情况下，就可以获得大部分AA的能力，甚至可以提供第三方代用户发起交易的能力，且不需要用户提供私钥，只需签名授权的信息。

### 4.2 数据结构

他定义了新的交易类型0x04，该交易类型的TransactionPayload 是下列内容的RLP编码序列化结果

重要的是其中新增了authorization\_list对象，存储签名者希望在其EOA中执行的代码，用户签署交易的同时也签署要执行的合约代码，他作为二维列表存在，说明可以批量存放多个操作信息，执行批量操作。

### 4.3 交易生命周期

4.3.1 验证阶段

在执行交易的开始阶段，对于每个authorization\_list的 [chain\_id, address, nonce, y\_parity, r, s] 元组：

从签名r、s中采用ecrecover恢复出签名者地址（注意这是以太坊本身的机制，所以该EIP没有改变签名算法）。authority = ecrecover(keccak(MAGIC || rlp([chain\_id, address, nonce])), y\_parity, r, s]（与之前解签名得出from地址类似，这里得出的是针对这个list的局部签名地址）

验证链ID（防分叉链重放）。

验证authority签名者的代码是否为空或已经委托（验证交易是否属于有效7702交易，后续会通过delegation机制去代理执行交易）。

验证authority签名者的nonce（防authority的签名重放）。

设置authority签名者的代码为 0xef0100 || address（用于绕过EIP3607防碰撞策略的）

增加authority签名者的nonce（防局部签名重放）。

将authority签名者账户添加到已访问地址列表中（转热地址，降低查询存储的gas费）

4.3.2 执行操作阶段

要执行的合约代码以及操作指令在哪里？

“新”版本仅更改了代码部署方面的行为。

它不再将帐户代码设置为contract\_code，而是从authorization\_list中检索代码address并将该代码设置为帐户代码。

所以，当需要执行授权代码时，从authorization\_list的 address字段指定的地址加载代码，并在签名者账户的上下文中执行。

这意味着用户的合约代码实际上是存储在链上的某个特定地址，而不是直接包含在交易中。

而操作指令和相关参数则存储在交易负载的data字段中。

### 4.4 EIP-7702有什么价值？

他对于Web3钱包的全链路都会有变化，用户体验也因此巨变，因为EOA发起的普通交易也可以类似合约执行多种逻辑，比如批量transfer。对于CeFi场景会影响交易鉴别，也影响冲提归集手续费

由于其出现，打破了很多曾经的定势，比如：

打破了账户余额只能因源自该账户的交易而减少的不变量。

打破了交易执行开始后 EOA nonce 增加1的不变量（可能同时增加多个）。

打破了tx.origin和msg.sender两个比对的防护逻辑，很多过往的合约有风险了。

打破了EOA本身无法发出事件的现状，对部分链上事件识别监听可能需要注意。

打破了EOA地址接受ERC20、721、1155等资产必然成功的现状（因为回调机制，可能失败）

### 4.5 对比EIP-7702和EIP-4337

1. EIP-7702的优势

gas更低，因为无需经过entrypoint模块，减少链上操作。

用户迁移成本更低，无需提前部署链上合约做为主体

与Eip4337相比，同样会有代码委托执行，也同样会有两种方式：

完全委托（Full Delegation）

完全委托是指将某个操作的全部权限委托给一个特定地址。例如，用户可以将所有ERC-20代币的管理权限委托给一个智能合约地址，从而使得这个智能合约可以代表用户执行所有相关操作。

受保护的委托（Protected Delegation）

受保护的委托是指在委托的过程中增加一些限制和保护措施，确保委托操作的安全性和可控性。

例如，用户可以仅将部分ERC-20代币的管理权限委托给一个智能合约，或者设置一些限制条件（如每天最多花费总余额的1%）。

2. EIP-7702的缺点

他的核心缺点是属于软分叉升级，需要大家共识推动，并且改动巨大，对链上生态影响太广，十四君初步评估下来，就有以下挑战，但是挑战也就是市场的机会：

自由度极高，难以被审计，用户会更需求靠谱的钱包承担安全防护的保护。

对原架构变化过大，虽然用不同交易类型区分，但是很多基建尤其链上不可改合约都无法直接适配。

对EOA地址提供了合约能力，但对应的存储空间无法留存。

单独交易的成本稍微提高，因为会大量增加Calldata的部分，估算调用的总成本将是16（gas） \* 15（字节） = 240（gas）calldata 成本，加上EIP-3860的成本2 \* 15 = 30，再加上大约 的运行时成本150。因此，仅仅准备账户哪怕什么都不做，就要增加500的Gas了。

“如果接收者签署了没有接收功能的代码，发送者在尝试发送资产时可能会面临 DoS。” 见案例。这个问题其实是 EOA A 签署了它不应该签署的东西——一个设置了错误实现（没有receive()）的可重放文件。

链上 冲提逻辑可能不一致，比如当转移 ERC-20 代币时，如果接收方账户有代码，则代币合约将调用onERC20Received接收方账户。如果onERC20Received还原或返回错误的值，则令牌传输将还原。

另外如果 EOA 可以发出事件，会不会有什么问题？一些基础设施可能需要注意。

这些还只是十四君基于目前EIP7702提案内容，以及对应的官方论坛讨论总结出的一些缺点，最终还需要基于最终的实现代码才能分析完全。

参考如下：

## 5、 全文总结

本文看似篇幅宏大，实际上文字内容只有6k余字，中间涉及的很多往期EIP解读，都在文中链接可以拓展，我就不进而追溯了。

目前看，账户抽象，确实只能放在第六模块，即修复一切，也即最后在推行，如今大幅加快EIP7702的进度，更多带来的还是对系统安全性的挑战，可以预料到，最终他会实现，毕竟以太坊合并，修改共识算法这样的颠覆性事件都可以发生，又谈何区区新的交易类型呢。

但是这一次颠覆的内容太多了，打破多个链上不可能的潜规则，也打破了大多数Dapp的应用逻辑，但是他死死的占住了最核心的一点，就是用户的成本更低了！对比EIP4337近乎翻倍的交易成本而言。

用户本身还是EOA地址，只是在需要的时候才去驱动和使用CA逻辑，所以持有成本低了。无需先转换出链上CA身份再做操作，等于用户无需注册了。

用户可以轻松用EOA做到多交易并行，比如授权代扣和执行代扣两种合一，这样对用户交易成本本身就低了，而对于Dapp而言，尤其是需要做链上企业级管理的项目方，比如交易所等更是颠覆性的优化，批量归集一旦原生态实现，基本交易所成本可以瞬间减少一半以上，最终也可以惠及用户。

所以，虽然他改变了很多，但占据成本这个维度，就值得全部Dapp去研究和适配，因为这一次，用户必然站在了EIP7702的一边。