账户抽象 AA

以太坊改进用户体验的探索

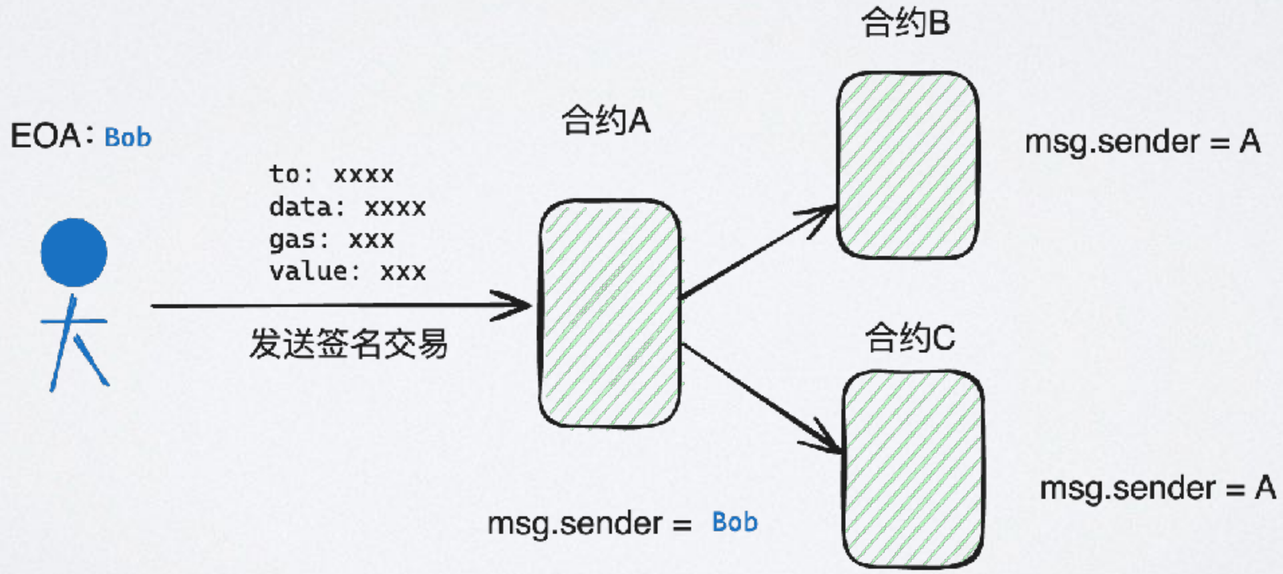
Tiny熊

要点

- 回顾 EVM 账户交易
- 以太坊在账户改进上的各种尝试
- ERC4337 DEMO 和 EIP 7702 实操

EVM 交易工作流

- 交易只能由 EOA (没有代码的账户)发起 EIP3607
- 但 EOA 和合约账户在 EVM 上是一样的，有同样的属性：balance、nonce、code、 state



EOA 特点

- EOA (Externally Owned Account 外部账号)
  - 用私钥控制的账号（MetaMask / ImToken … ）
  - 特点：链下生产、所有链账号一致
- 问题：
  - 私钥、助记词、支付 Gas（普通用户太不友好）
  - 一次只能签一笔交易
  - 无法 Social Recovery、无法替换 Key

要大规模采用，不可能

交易体验不佳

```
// ERC20
    function approve(address _spender, uint256 _value) {
        allowance[msg.sender][_spender] = _value;
        return true;
    }
```

```
// DEX
    function swap( ) {
        token.transferFrom(msg.sender, address(this), amount)

    }
```

合约账户（CA）

- 由合约代码控制的账户
- 逻辑灵活：其他签名方式（如 Passkey secp256r1）、多签、Social Recovery、Gas 代付
- 但是只能被动执行

能否尝试融合两者： 账户抽象（Account Abstraction）

先驱：EIP-86 / EIP-2938

- EIP-86：抽象验证签名逻辑，创建“合约”来验证交易签名
- EIP-2938：让合约成为一等公民（可支付 Gas 及发起交易）

对核心协议修改太大，暂时搁置

ERC- 4337

- 不改变协议（不影响共识）
- 合约作为账户（**Smart Account**），引入 Bundler 打包发送交易
- 带来了前所未有的灵活性：
  - 多种签名逻辑 （在合约中验证签名）
  - 批量交易
  - 社交恢复
  - 用其他 Token 代付手续费，Paymaster 作为付款人

在一些 layer2 上原生支持 AA（RIP-7560）： Starknet、zkSync 等

ERC-4337 案例

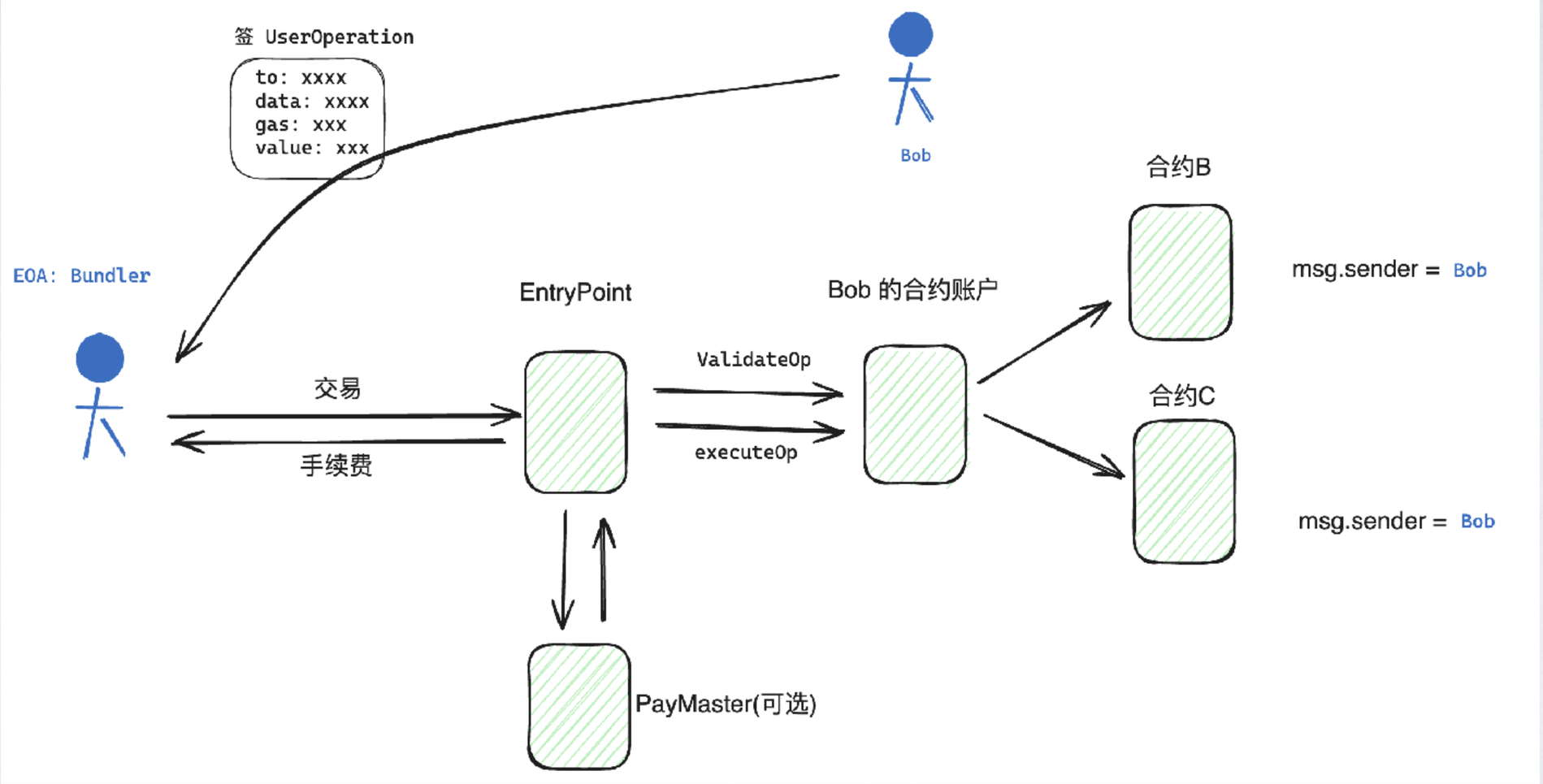
利用 Passkey 迁移传统 Web2 用户到链上

- 基于 Passkey 方案为用户创建账户，让用户不用关注助记词和私钥。
  - Passkey 是基于FIDO2/WebAuthn标准实现的无密码登录，使用 secp256r1 (P-256) 曲线在用户设备上签名（表现为指纹、faceID 等），服务端验证登录，主流的 Web 服务均支持
- 在合约中对验证 secp256r1 签名（RIP-7212 已经引入 P-256 预编译），即可实现合约钱包，该钱包由 Passkey 签名控制

ERC-4337 Demo

https://passkey-demo.zerodev.app/

ERC-4337 交易工作流



ERC-4337 主要组件

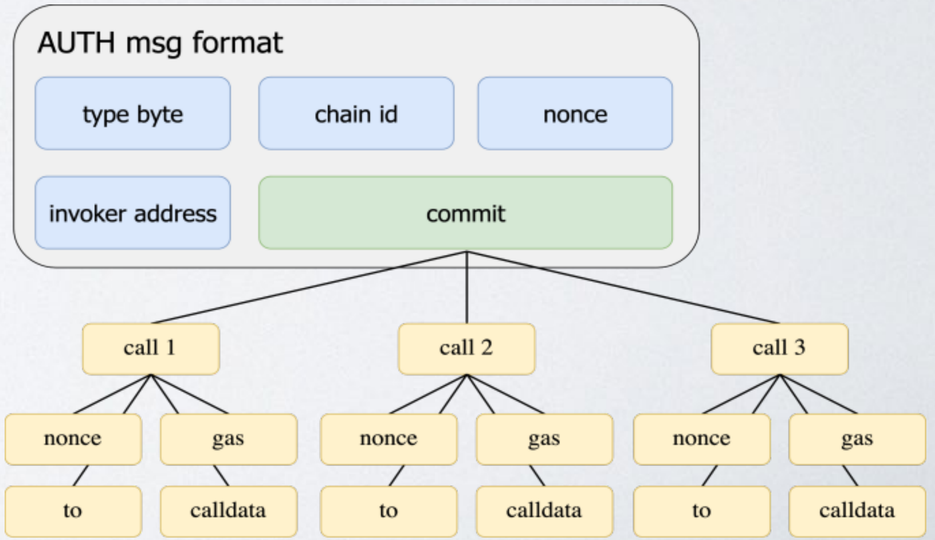
- Smart Contract Wallet: 验证和执行 UserOp 中的 callData , 可以有自定义的验证逻辑
  - 通常由 Account Factory 创建
- Bundler： 将用户的 UserOp 发送到链上 EntryPoint ，
- EntryPoint： 单例智能合约， 用来验证（Validation）和执行（Execution）UserOp（对应着原来客户端的实现）
- PayMaster （可选）： 代付手续费

ERC-4337 基础设施（服务商）

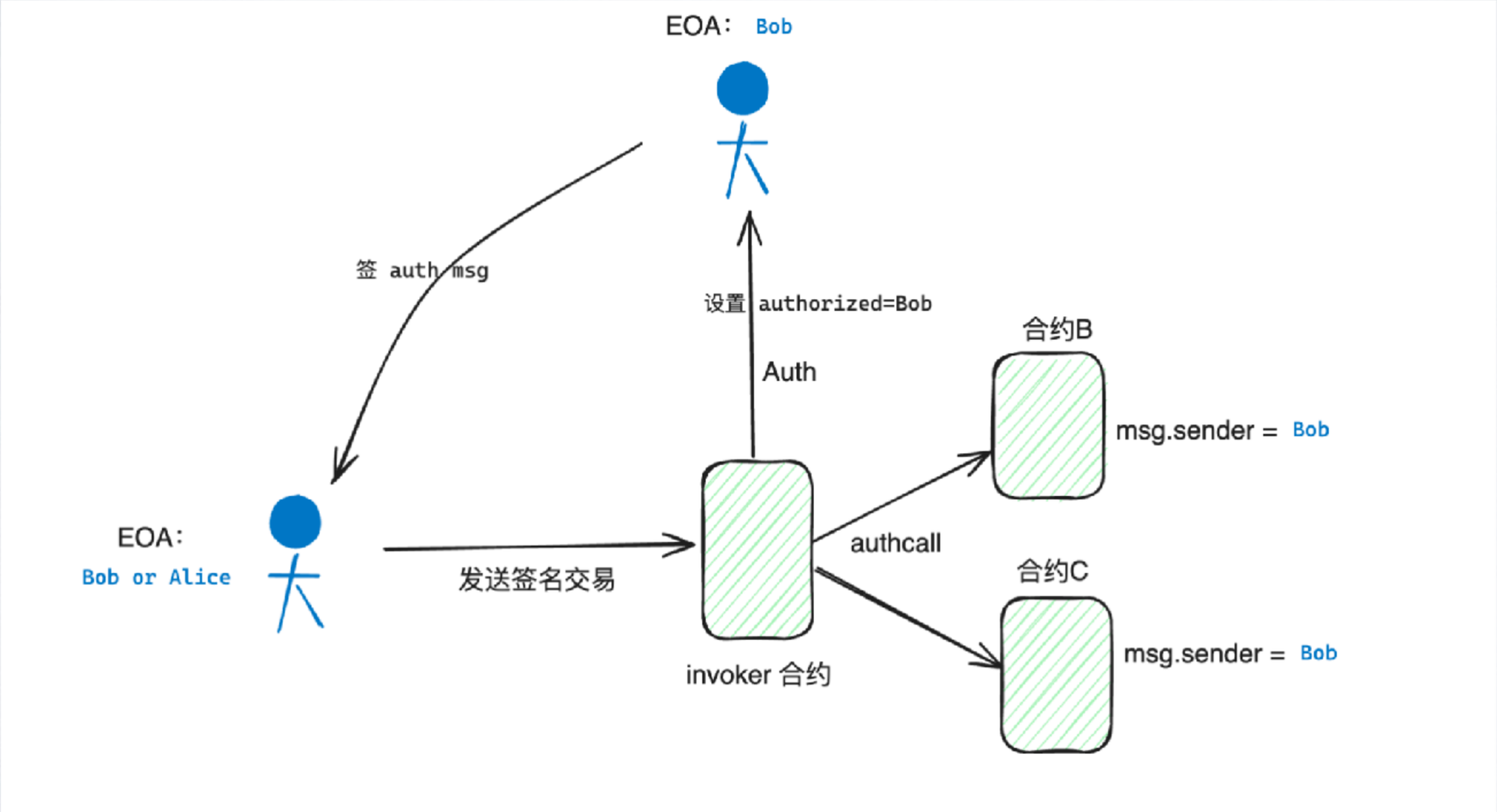
- SDK：Zerodev 、 Safe Wallet 、Pimlico、biconomy、etherspot、Alchemy AccountKit 、thirdweb
  - 使用这些 SDK 方便在自己的应用里集成钱包
- 理解 ERC4337 实现参考文章：
  - https://learnblockchain.cn/article/5426
  - https://learnblockchain.cn/article/5432
  - https://learnblockchain.cn/article/5442
  - https://learnblockchain.cn/article/5483

Eip-3074 - 重新回到视野

- 创建账号成本高、交易费高、多链隔离，让 ERC-4377 采用率不够理想。
- EIP-3074 引入操作码：
  - AUTH : invoker 合约中设置 authorized 变量为授权签名的账户地址
  - AUTHCALL : 用 authorized 变量作为调用者地址，再进行 Call 调用
- 允许智能合约代表 EOA （为现有账号添加功能）
- 可实现 Gas 代付、批量交易



EIP-3074 工作流


EOA: Bob
签 auth msg
设置 authorized=Bob
Auth
合约B
msg.sender = Bob
EOA:
Bob or Alice
发送签名交易
authcall
合约C
msg.sender = Bob
invoker 合约

EIP-3074 争议

- 优点：
  - Gas 低
  - 可快速应用（利用现有的 EOA及基础设施）
- 反对：
  - Invoker 权力过大，不利于抗审查
  - 依旧依赖 ECDSA
  - 在 EOA 到 SA 迁移后（被认为是 endgame），EIP-3074 引入的操作码将无用，但会遗留在 EVM 代码中，造成技术债务。

EIP-7702

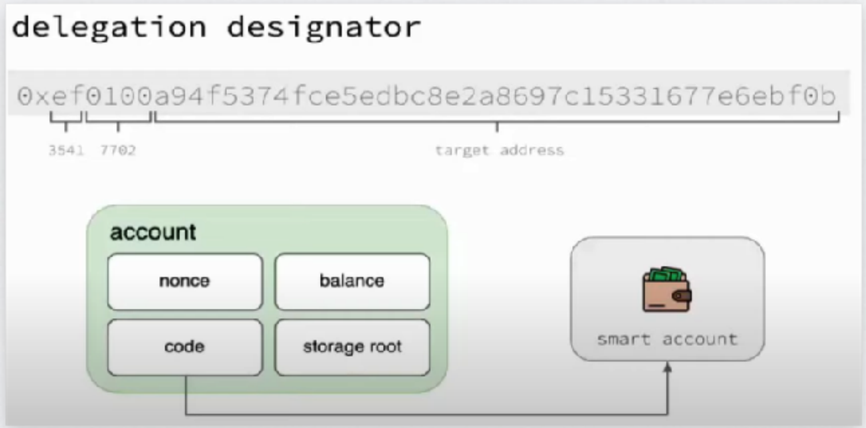
- 作为 EIP-3074 的替代方案提出
  - 向后兼容现有钱包，向前允许转为合约钱包（可兼容 ERC4337 ）
  - 实现批量交易、Gas 代付，但不引入技术债务
- 添加新的交易类型，而不是添加指令设置代码，以便让 EOA 可以 delegatecall 合约账户（从而让 EOA 变成 smart wallet）
  - 具体是在交易中加入一个 authorization_list
  - authorization_list 为 [chain_id, address, nonce, y_parity, r, s]

委托的合约地址

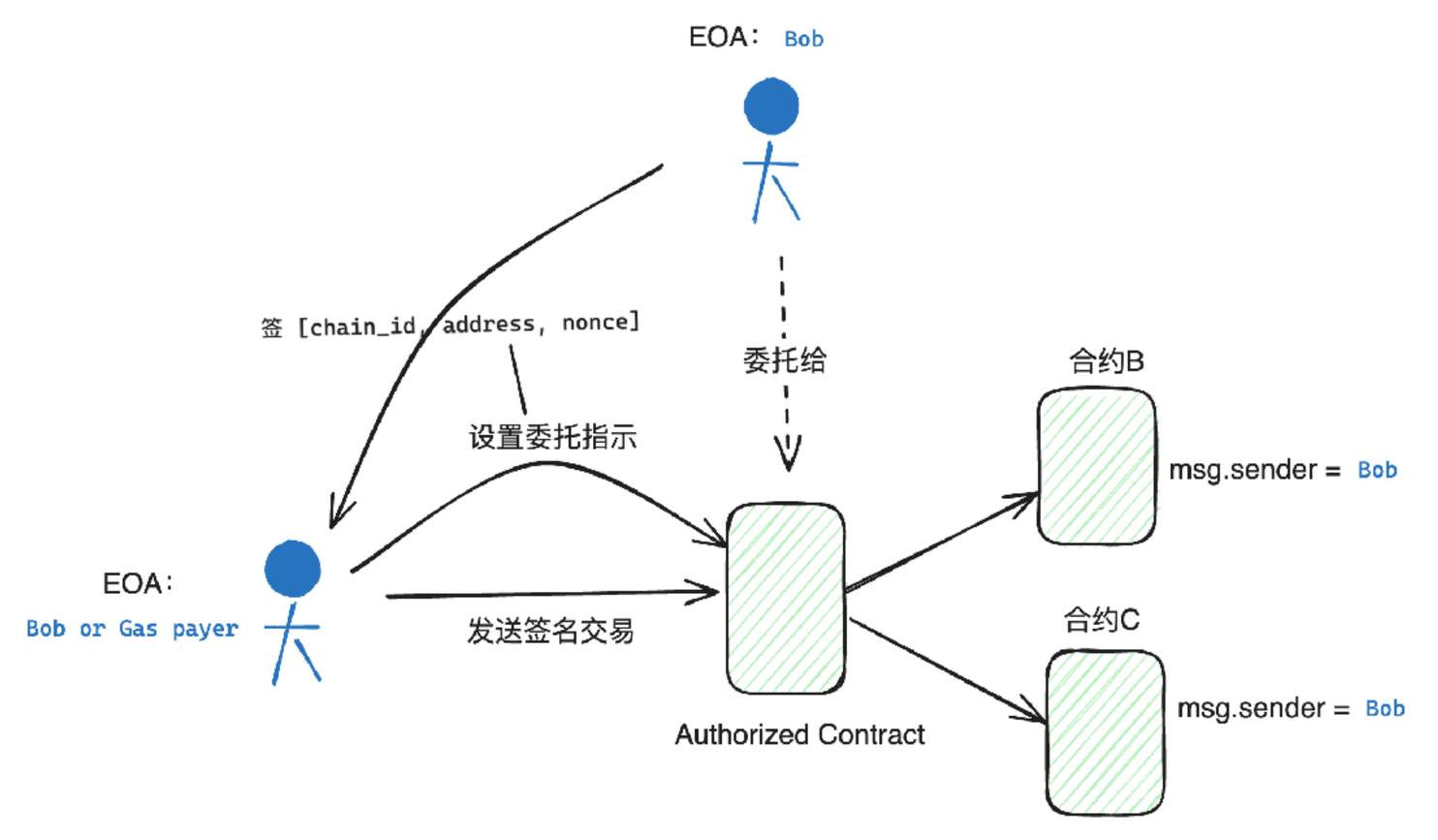
```
chain_id,
nonce,
max_priority_fee_per_gas,
max_fee_per_gas,
gas_limit,
destination,
value,
data,
access_list,
"authorization_list",
signature_y_parity,
signature_r,
signature_s
```

EIP-3607 有话说

- 兄弟，有合约代码的账户， 不可以发起交易（代码为空才可以发起交易）。
- EIP7702 为 EOA 设置了一个特殊的代码：Delegation Designator， 以 0xef0100 开头，不被认为是有效合约代码（EIP-3541）。
- EIP-3607 行为被修改： 空代码 或 0xef0100 的开头代码都可以发起交易

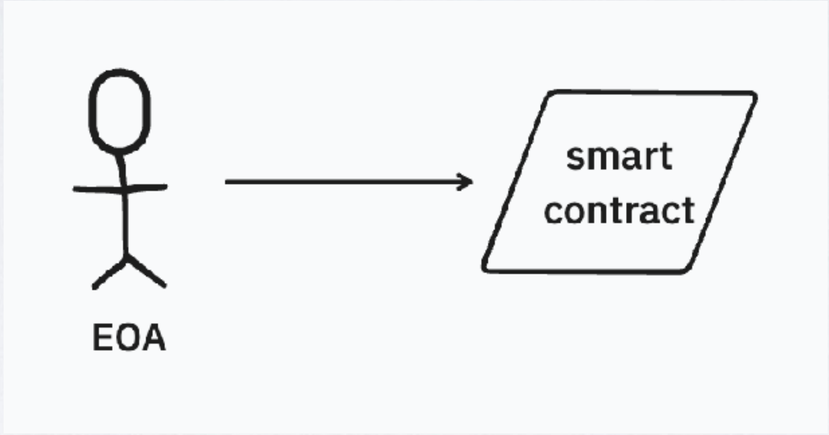


EIP-7702 工作流

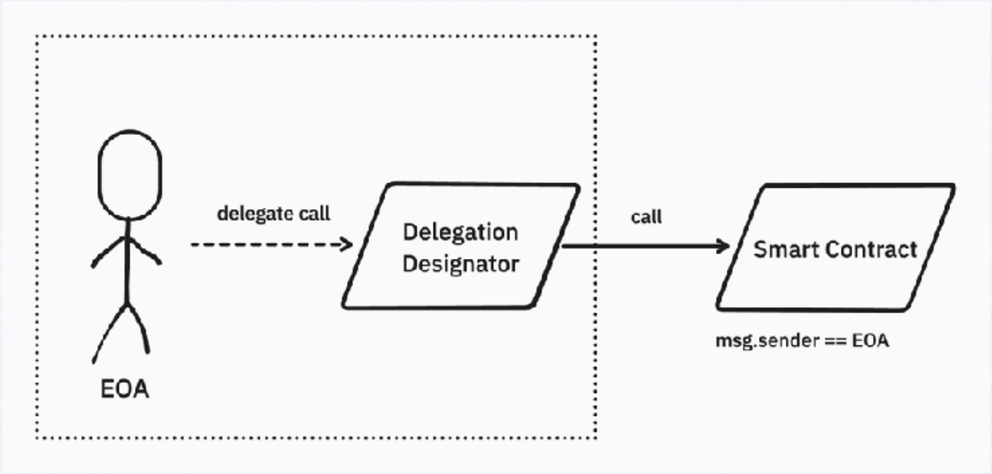

EOA: Bob
签 [chain_id, address, nonce]
设置委托指示
委托给
EOA:
Bob or Gas payer
发送签名交易
Authorized Contract
合约B
msg.sender = Bob
合约C
msg.sender = Bob

Eip-7702 账户多态性

作为 EOA

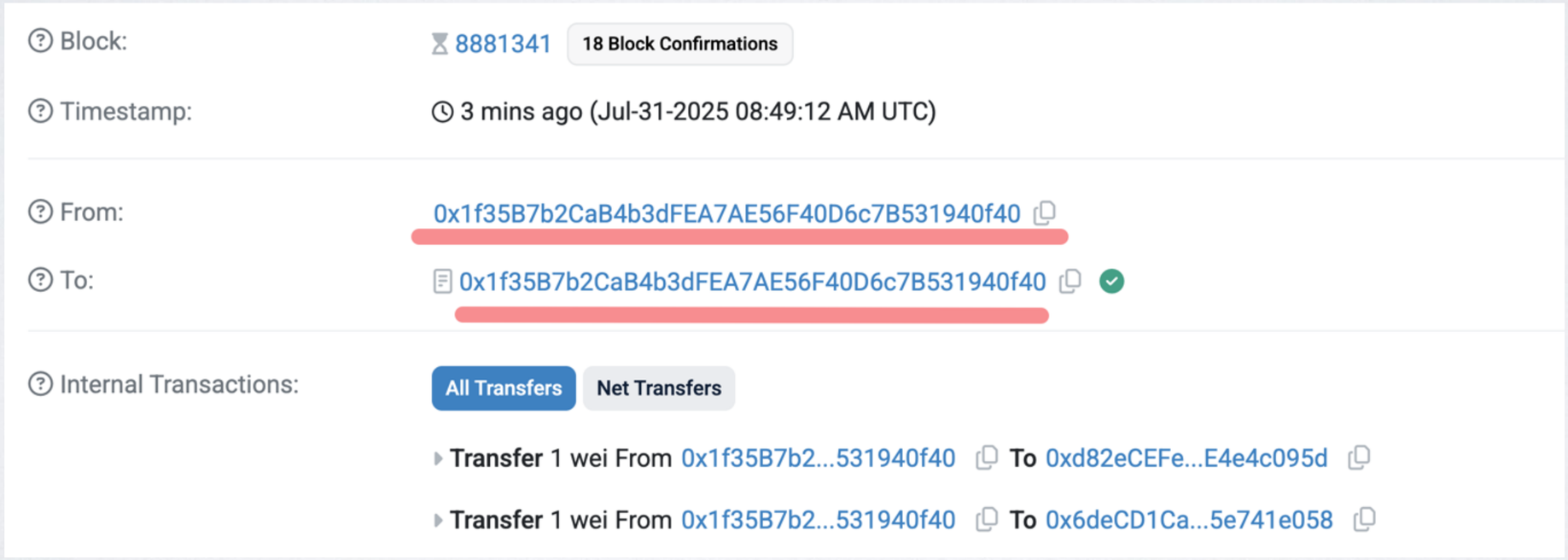


作为合约
(可批量调用其他合约)



EIP7702 - Demo

https://7702-readiness.vercel.app/



Eip-7702 实践

- 使用 7702 一次发送两笔交易

1. 签名授权给右侧合约
2. 调用execute() 在参数中封装两个调用
   ① approve()
   ② deposit()

```
contract SimpleDelegateContract {

    function execute(Call[] memory calls) external payable {
        for (uint256 i = 0; i < calls.length; i++) {
            Call memory call = calls[i];
            (bool success, bytes memory result) = call.to.call{value: call.value}(call.data);
            require(success, string(result));
            emit Executed(call.to, call.value, call.data);
        }
    }
}
```

合约：https://github.com/lbc-team/hello_foundry/blob/main/src/SimpleDelegateContract.sol

viem 调用：https://github.com/lbc-team/hello_viem/blob/main/demo/src/build_7702_tx.ts

Eip-7702 还可实现

- 批量交易
- 交易赞助
- 将 EOA 转为多签控制
- 设置社交恢复密钥
- 其他的签名方式控制
- 指定生命周期的支付
- …

7702 委托合约参考实现

https://www.bundlebear.com/eip7702-authorized-contracts/all

- https://github.com/MetaMask/delegation-framework/blob/v1.3.0/src/EIP7702/EIP7702DeleGatorCore.sol
- https://docs.openzeppelin.com/community-contracts/0.0.1/eoa-delegation
- https://github.com/zerodevapp/kernel
- https://github.com/ithacaxyz/exp-0001/blob/main/contracts/src/ExperimentDelegation.sol
- https://github.com/okx/wallet-core

使用 7702 应注意

- msg.sender == tx.origin 限制 EOA 账户方法失效，不能用来此方式来限制防御重入攻击。
- 使用 7702 要尽量避免向合约存储写入数据，防止替换委托时，产生存储冲突，如何需要写入，应遵循 ERC-7201 提出的 Namespace 隔离存储。
- 一些合约要求接收者实现回调， 要注意与这些合约兼容，例如 ERC-721，ERC-777

总结对比

ERC-4337	EIP-7702
无协议变更	引入新交易类型
创建独立的合约账户	（瞬时）给 EOA 添加代码
需要资产迁移	复用现有资产
可不依赖 ECDSA	依赖 ECDSA
Gas 较高	Gas 较低

作业（可选）

- EIP 7702实践：发起打包交易
- 部署自己的 Delegate 合约（需支持批量执行）到 Sepolia。
- 修改之前的 TokenBank 前端页面，让用户能够通过 EOA 账户授权给 Delegate 合约，并在一个交易中完成授权和存款操作。

https://decert.me/quests/2c550f3e-0c29-46f8-a9ea-6258bb01b3ff