开发自己的钱包

EOA钱包 - CLI

合约钱包

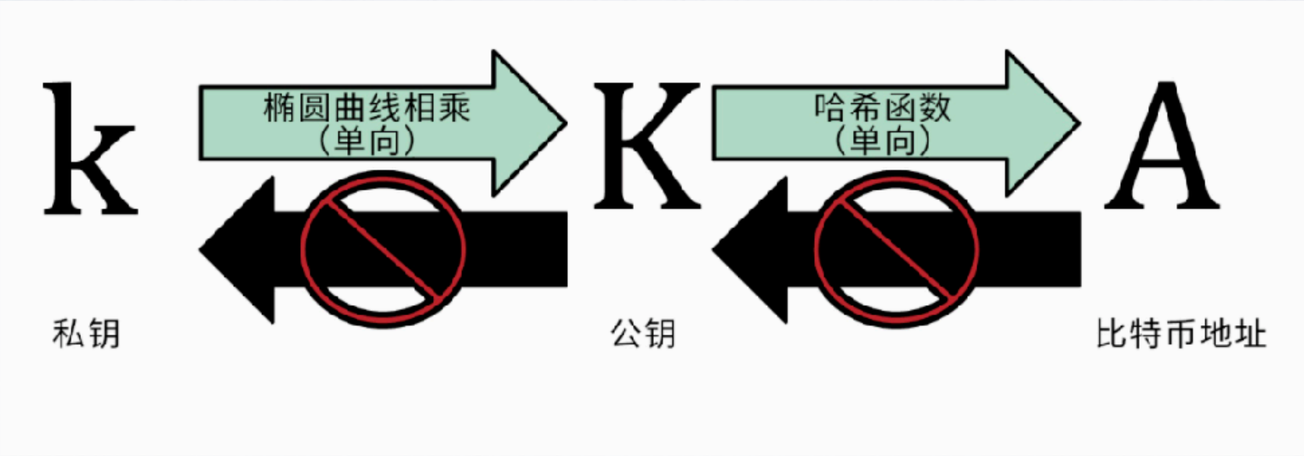
（EOA）钱包核心功能

- **账户管理：创建账号、导入导出**
  - **私钥、助记词、Keystore、MPC**
- **构造交易、签名、发送交易**
- 其他：展示资产价值、交易记录、钱包浏览器、资产交易等

EOA 账号创建

EOA 账号

- 本质是一个 32 字节的私钥（随机数）



Demo: OpenSpace100/account_demo/create_by_raw.js

创建 EOA 账号

- 找到一个安全的熵源（不可预测、不可重复）作为私钥，如掷硬币256次。
- 使用 secp256k1椭圆曲线算法计算出公钥
  - secp256k1 : 比特币和以太坊 $y^2 \equiv x^3 + 7 \pmod{p}$
  - Secp256r1: Passkey SSH $y^2 \equiv x^3 - 3x + d \pmod{p}$
  - Ed25519 : Solana $-x^2 + y^2 \equiv 1 + dx^2y^2 \pmod{p}$
- 对公钥进行 keccak256 hash运算再取后40位得到

Demo: OpenSpace100/account_demo/create_by_raw.js

管理私钥的几种方式

- 随机生成的私钥，备份麻烦，不易管理
- 通常采用以下几种方式来管理：
  - 助记词（分层确定性推倒）
  - Keystore
  - MPC

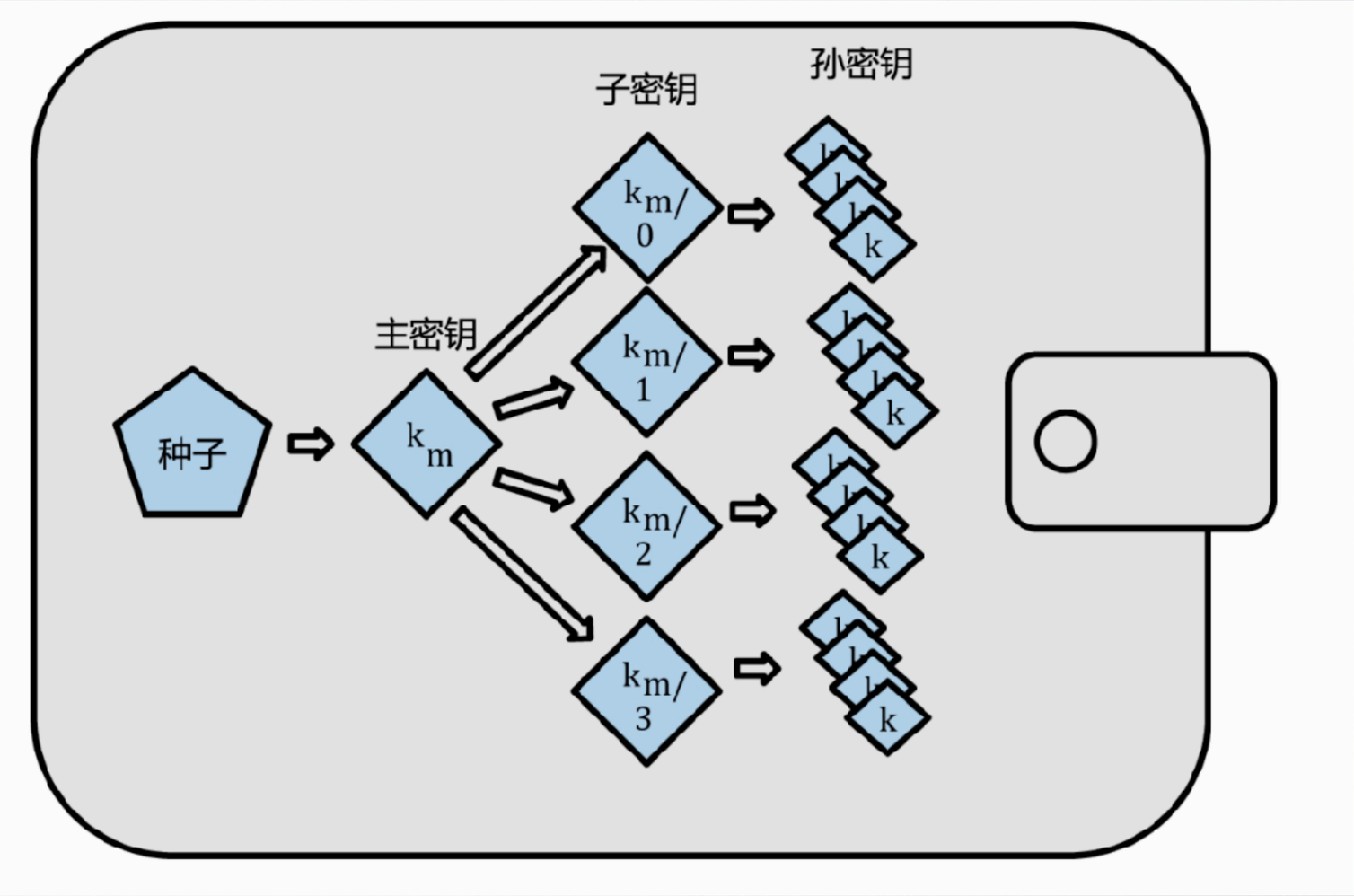
分层确定性推倒

- 随机生成的私钥，备份麻烦，不易管理
- BIP32 提案（HD钱包）：由同一个种子，就可以生成无数个私钥和地址
- BIP44 提案： 给bip32的路径赋予意义来支持做多币种、多地址
- BIP39: 使用助记词的方式，生成种子

BIP32

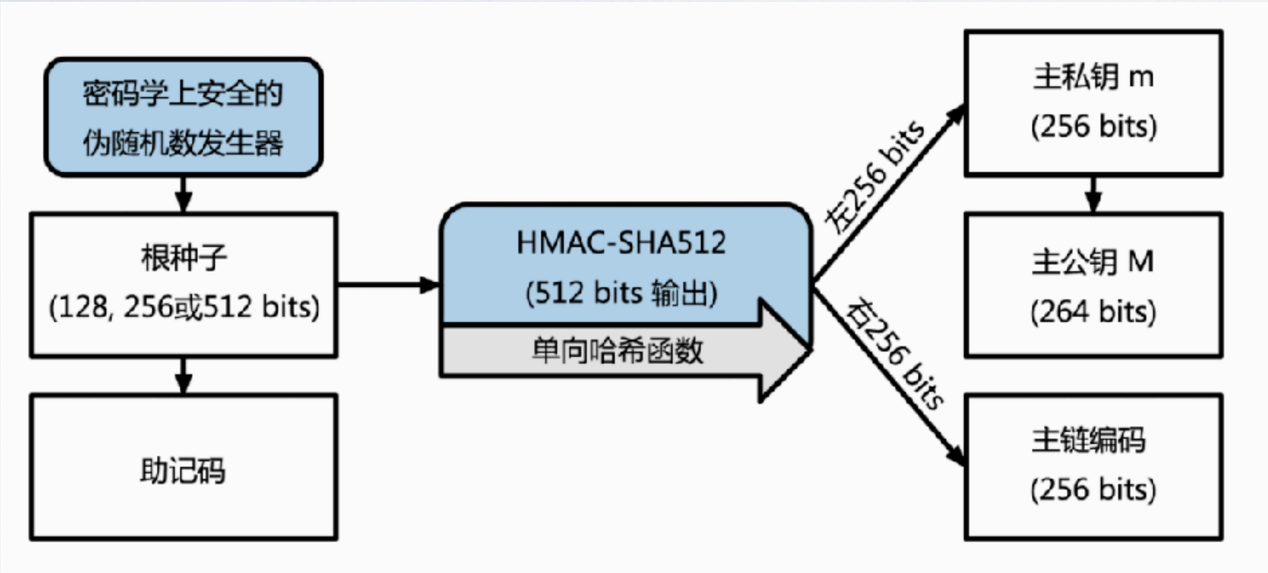
种子推倒生成私钥

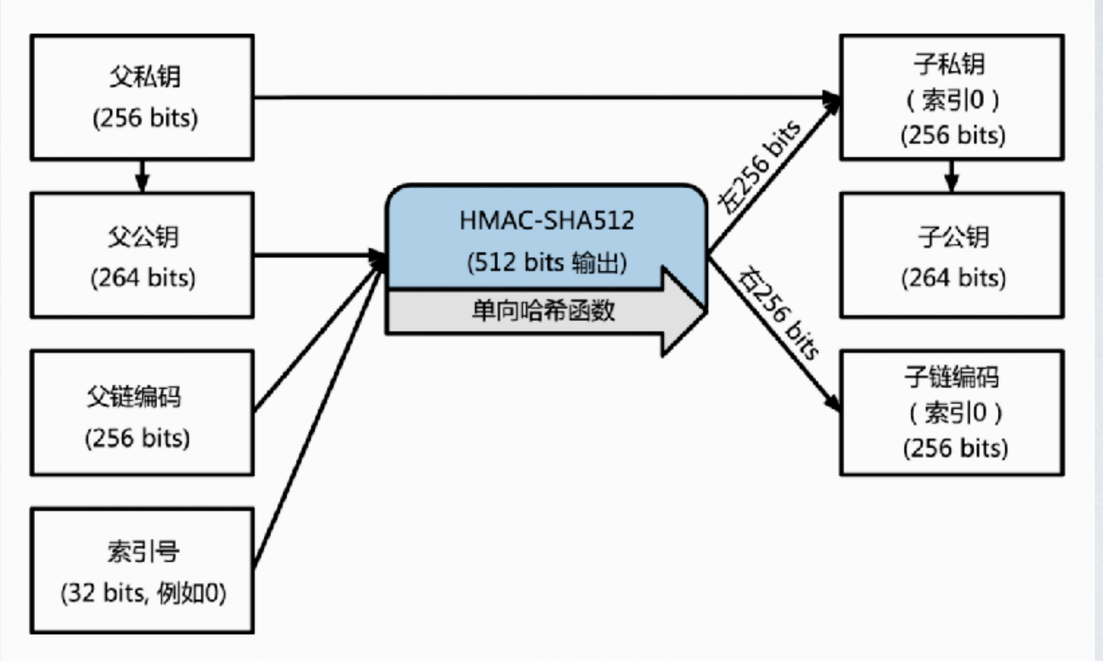
https://learnblockchain.cn/2018/09/28/hdwallet/



BIP32 / BIP44

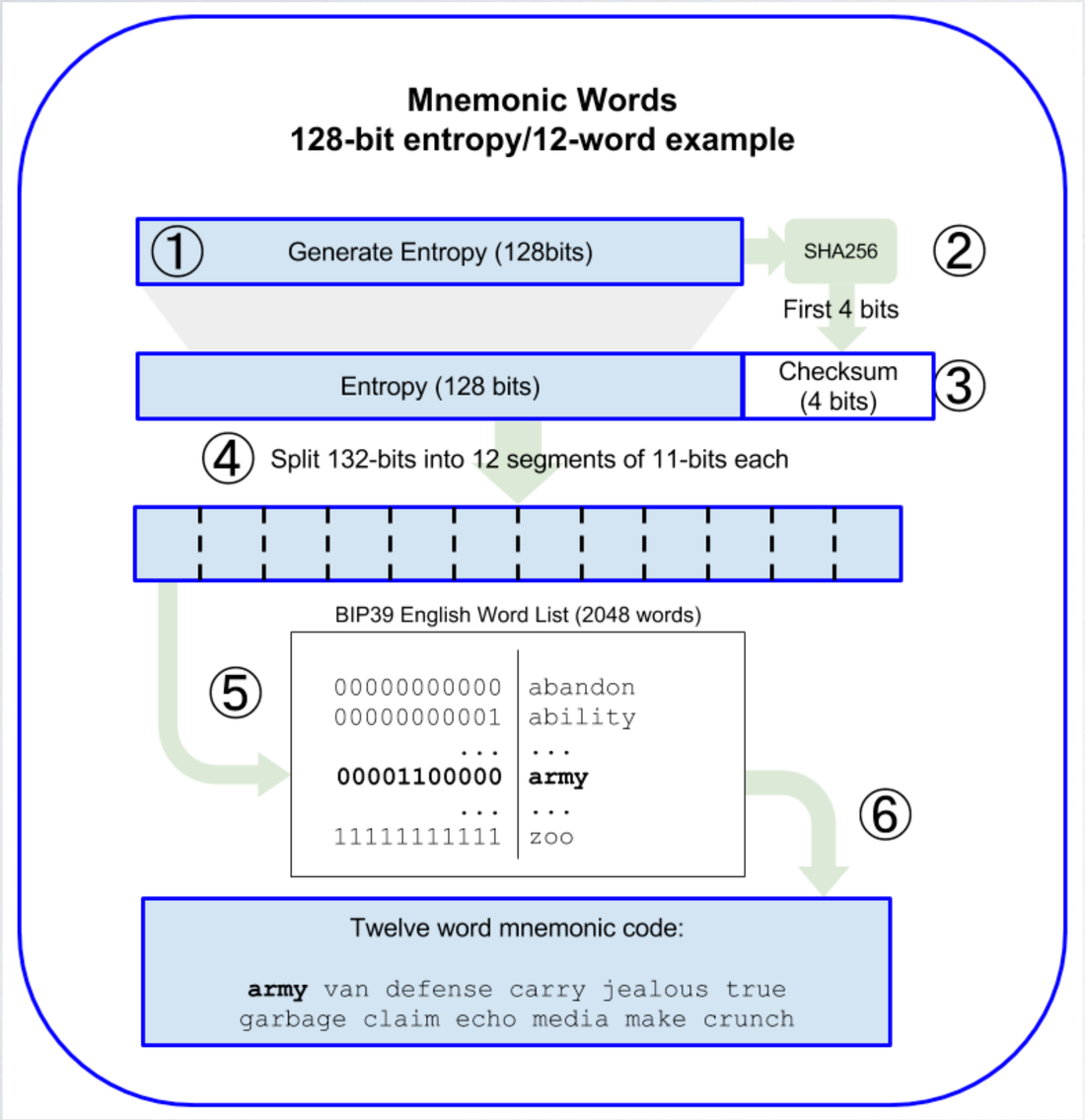
m/ I

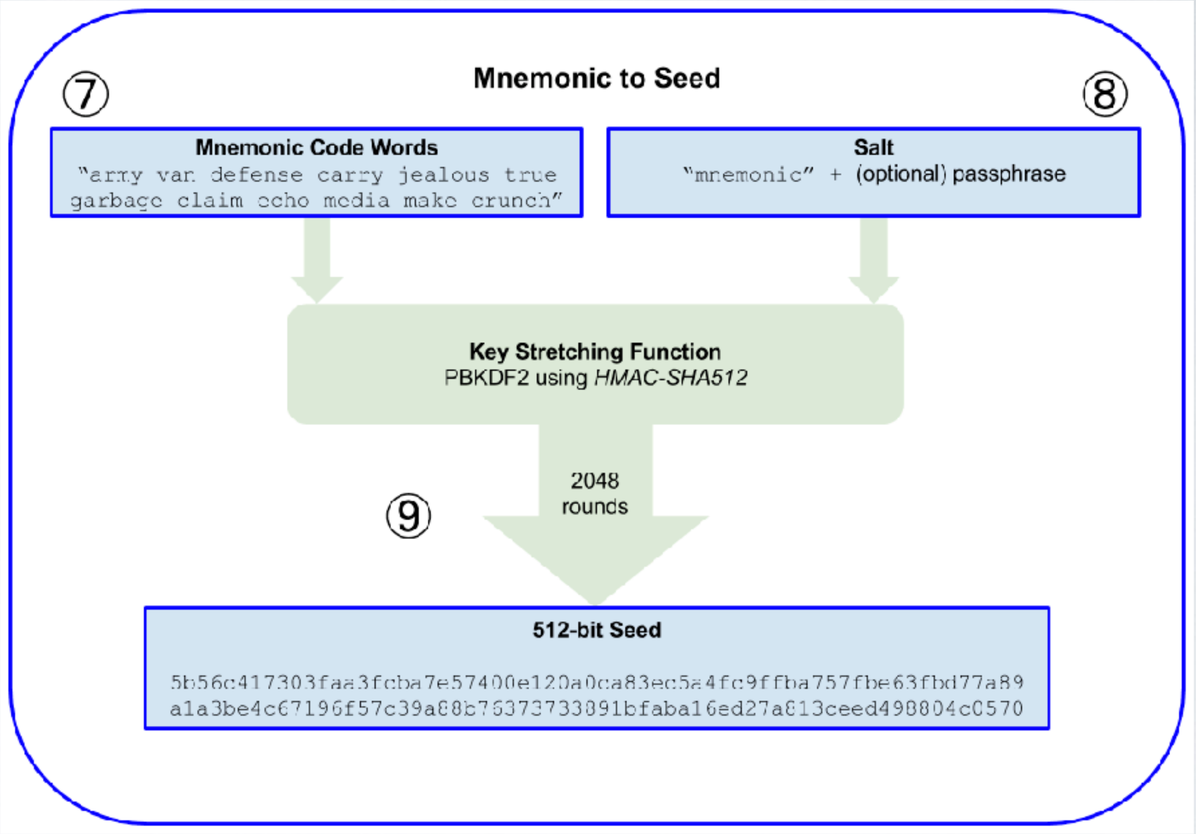




BIP44: m / purpose' / coin' / account' / change / address_index

BIP39

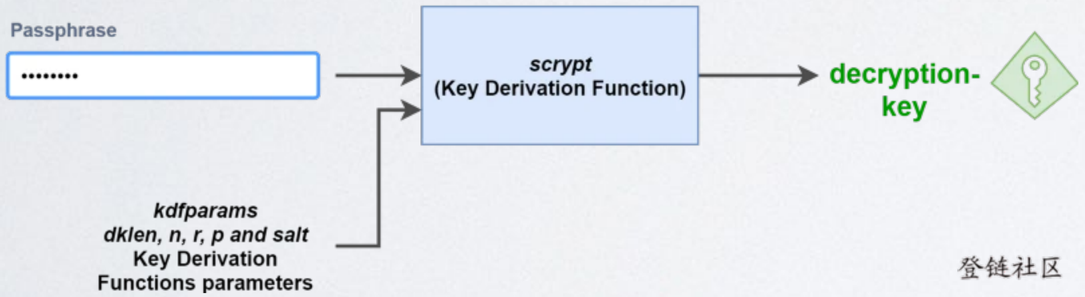


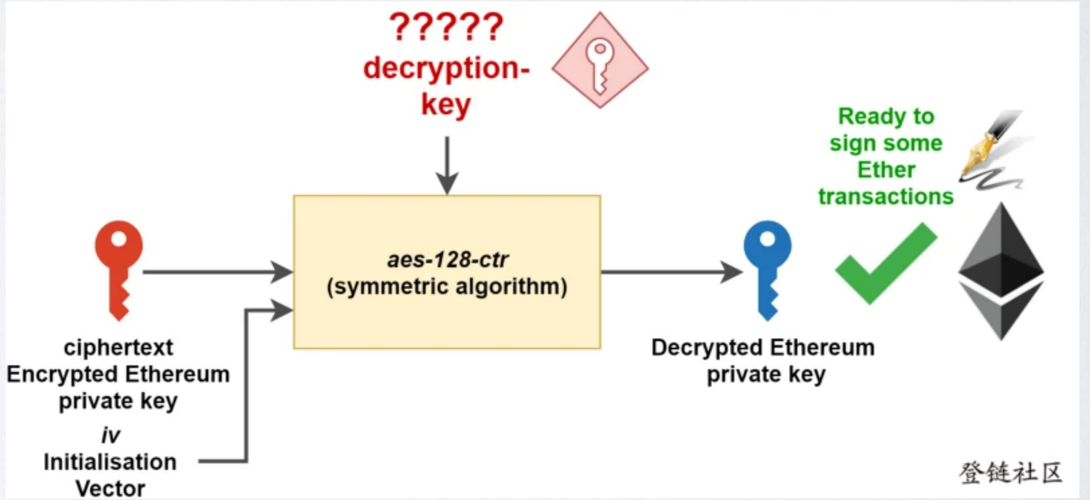


Demo: OpenSpace100/account_demo/create_by_mnemonic.js

KEYSTORE 加密私钥

- Keystore 文件就是一种以加密的方式存储密钥的文件（加强安全性），签名前先解析出私钥。
- 原理： 使用 scrypt 算法从密码派生加密密钥， 用加密密钥使用 AES-128-CTR 算法加密私钥。





https://github.com/lbc-team/hello_viem/blob/main/demo/src/keystore_demo.js

KEYSTORE DEMO

- Keystore 生成及校验
  - https://github.com/lbc-team/hello_viem/blob/main/demo/src/keystore_demo.js
- Keystore 发起交易
  - https://github.com/lbc-team/hello_viem/blob/main/demo/src/build_tx_keystore.ts

MPC 私钥分片

- MPC（Multi-Party Computation）多方计算
- 利用 MPC 技术可将私钥分片，每个分片由不同的参与方保管，通过 m/n 个分片聚合出私钥（或签名）。
  - SSS （ Shamir's Secret Sharing ）： 聚合后签名
  - TSS（ Threshold Signature Scheme ）： 聚合签名 ， 大多数项目采用的方案
- 部分分片泄露，私钥仍然安全，降低了单点故障风险

https://github.com/lbc-team/hello_viem/blob/main/demo/src/mpc_tx.js

MPC 钱包

- OKX Wallet： 2/3 MPC TSS 方案
  - OKX Cloud 分片
  - 用户设备分片
  - Google Cloud、iCloud 等云备份
- Para MPC 方案：利用 passkey 为 Dapp 应用提供**嵌入式钱包**体验， 2-2 MPC方案
  - 用户设备分片、Passkey 云同步（Google Cloud、iCloud 等可跨设备）
  - Para Cloud Share （用户可额外备份）
- 其他的钱包服务商：Dynamic、Turnkey 和 Privy （Stripe 收购）

构造交易

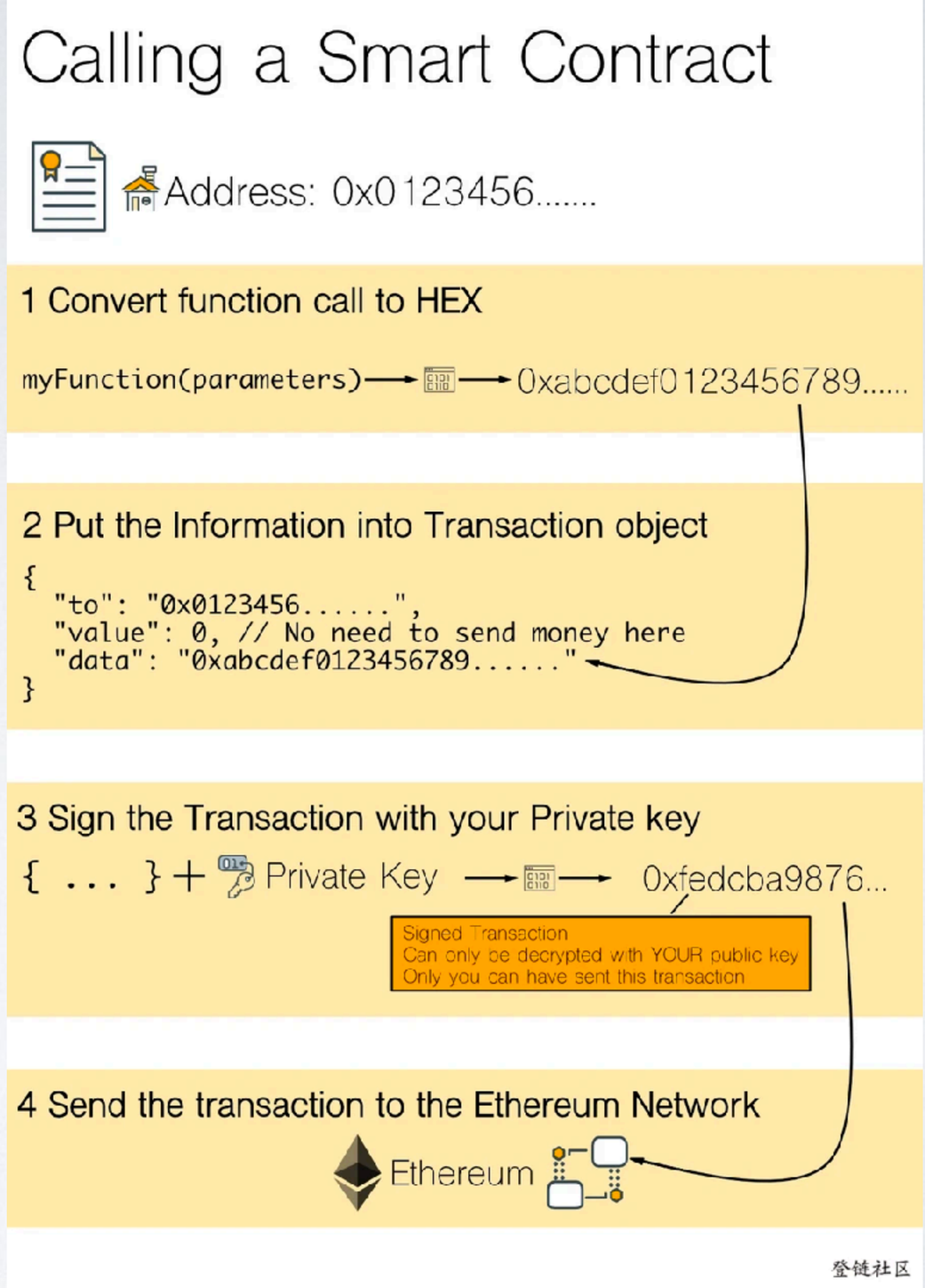
1. ABI 编码

2. **构造交易**

3. RLP 序列化、 Hash、**签名**

4. 广播上链

rpc： eth_sendRawTransaction



1. ABI 编码

- 函数选择器 + 参数编码
- 函数选择器：函数签名的 Keccak 哈希的前 4 字节
- 参数编码：第5个字节开始，基本类型扩容到 32 字节表示，复杂类型用起始位置、数据大小、真实数据

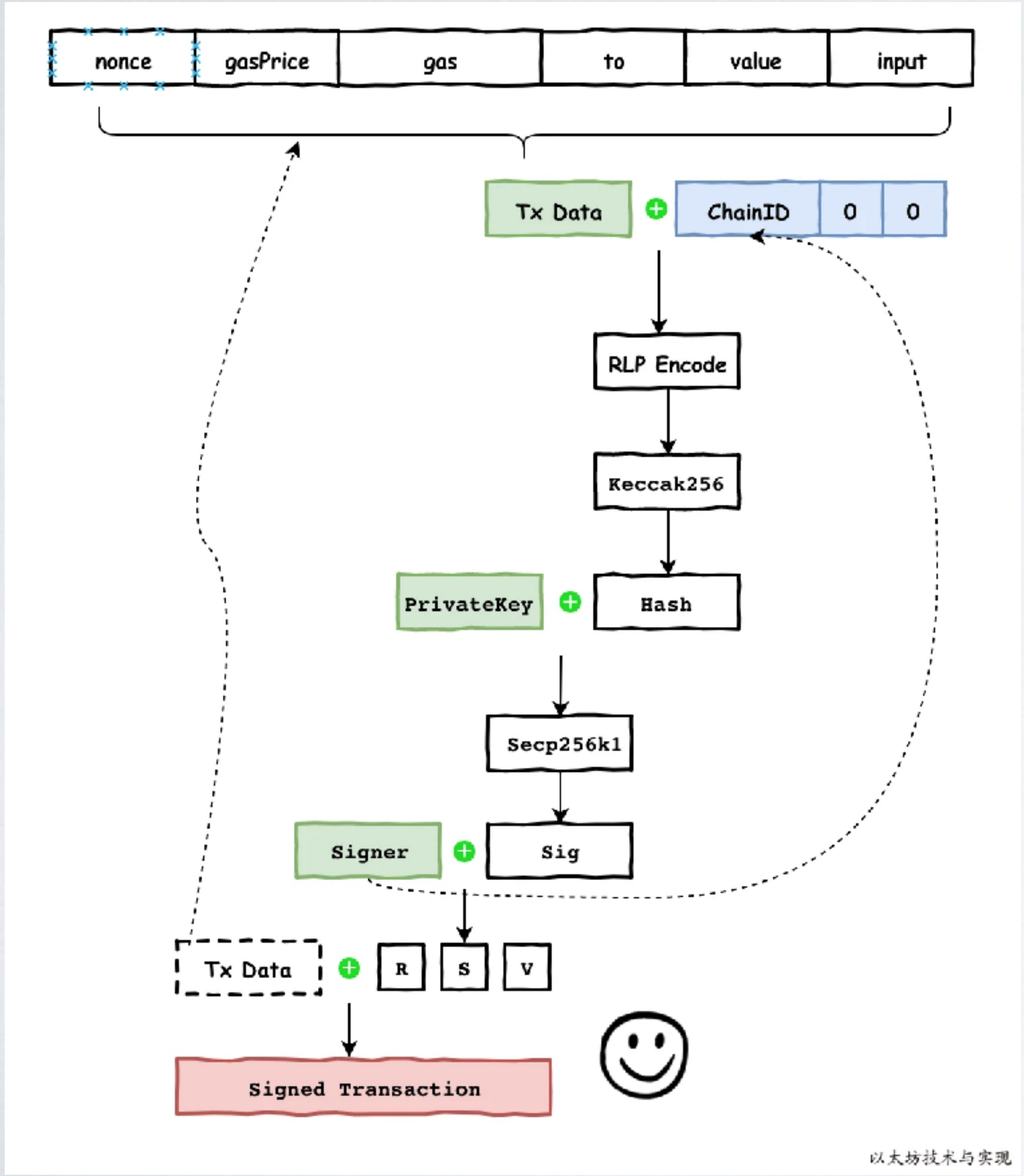
工具：

- Foundry: cast abi-decode / cast abi-encode
- Solidity: abi.decode / abi.encode
- Web3.js / ethers.js / viem
- https://chaintool.tech/calldata

2. 构造交易

```
Transaction {
  to: Address    // 交易的接收者
  nonce: Hex     // 发送者的nonce（已发送交易数量的计数）
  type: Hex      // 交易类型，0(legcy) 或 1(EIP-2930) 或 2(EIP-1559)
  value: Hex     // 交易携带的主币数量，单位是 wei
  data: Hex      // 交易携带的数据 （ABI 编码）
  maxPriorityFeePerGas?: Hex // EIP-1559:每单位 gas 优先费用，type=2时提供
  maxFeePerGas?: Hex         // EIP-1559:每单位 gas 最大费用，type=2时提供
  gas: Hex                   // 可使用的最大 gas 数量(gasLimit)
}
```

3. RLP 序列化、HASH、签名




```
[chainId, nonce, maxPriorityFeePerGas, maxFeePerGas, gas, to, value, data, accessList]
```

4. 广播交易、节点验证、执行打包

- RPC 节点收到交易
- 验证：签名正确、确保交易费够、GasLimit 不超限制、chainID 匹配、Nonce 是否符合预期
- 放入交易池（mempool）
- 从交易池取交易（一般按交易手续费排序）、EVM 执行

完整的交易流程

- 创建私钥（或助记词、或导入）
  - 可选：查询余额、充值
- **构造交易对象：{to: ， data:， value: , gas: …. , nonce: …}**
- 签名交易
- 发送交易到节点

```
await txParams = {
      account: account,
      to: '',
      value: parseEther('0.0001'), // 发送金额（ETH）
      chainId: sepolia.id,

      // EIP-1559 交易
      maxFeePerGas: parseGwei('40'), // 最大总费用（基础费用+小费）
      maxPriorityFeePerGas: parseGwei('2'), // 最大小费

      gas: 21000n,   // 普通交易 - gas limit
      nonce: 1,
    }

const signedTx = await walletClient.signTransaction(txParams)

const txHash = await publicClient.sendRawTransaction({
      serializedTransaction: signedTx
    })
```

DEMO 代码：https://github.com/lbc-team/hello_viem/blob/main/demo/src/build_raw_tx.ts

插件（MetaMask）钱包
怎么实现的…

METAMASK 如何签名的

- 通过插件在浏览器中注入了 window.ethereum
- 获取账号
- 发送交易

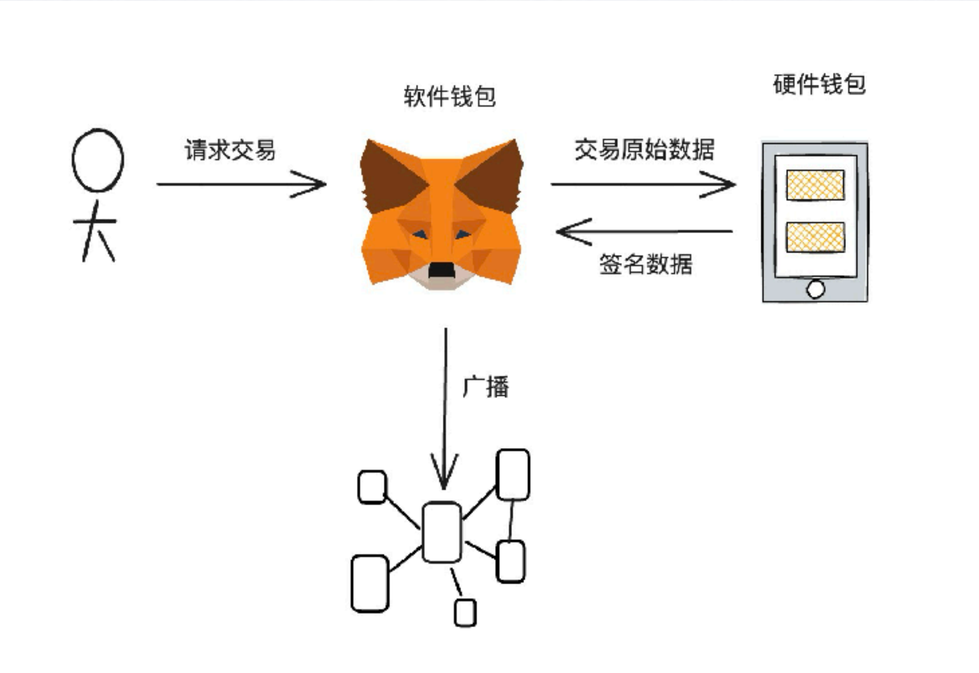
```
await window.ethereum.request({
 "method": "eth_accounts",
 "params": [],
});
```

```
await window.ethereum.request({
  "method": "eth_sendTransaction",
  "params": [
    {
      "to": "0x4B0897b0513FdBeEc7C469D9aF4fA6C0752aBea7",
      "from": "0xfc03347e0168838eb029049f98a35b14bd612c40",
      "gas": “0x76c00",
      "value": "0x0100",
      "data": “0x…”,
      "maxFeePerGas": "0x4a817c800"
    }
  ]
});
```

硬件钱包

硬件钱包

- “硬件钱包”和“冷钱包”常常被提及在一起：
  - 冷钱包：一种**离线存储私钥的方式**，强调的是“不联网”
  - 硬件钱包：是一种具体的冷钱包设备，强调的是“物理硬件”，需要配合软件钱包一起使用
- 硬件钱包： 安全芯片（存储私钥，防物理攻击、签名）、用 USB 或蓝牙通信签名信息与结果。



签名机

- 业务中，尽量保持私钥的服务器是隔离的，可以使用签名机 或 KSM 系统

作业：使用 VIEM 构建一个 CLI 钱包

- 编写一个脚本（可以基于 Viem.js 、Ethers.js 或其他的库来实现）来模拟一个命令行钱包，钱包包含的功能有：
  - 1. 生成私钥、查询余额（可人工转入金额）
  - 2. 构建一个 ERC20 转账的 EIP 1559 交易
  - 3. 用 1 生成的账号，对 ERC20 转账进行签名
  - 4. 发送交易到 Sepolia 网络。

https://decert.me/quests/992dae0f-3bdf-4f03-9798-3427234fad95

合约钱包

合约钱包 -> 智能合约钱包账号

- 外部账户（EOA）与 合约账户在 EVM 层面是等效的，都是有：nonce（交易序号）、balance（余额）、storageRoot（状态）、codeHash（代码）
- 如果该合约可以持有资金、调用任意合约方法，就是一个智能合约钱包账户
- 智能合约钱包账户：支持多签、multicall、密钥替换、找回
- AA（ERC4337 / ERC7702）： 抽象掉 EOA 与 智能合约钱包账户的区别

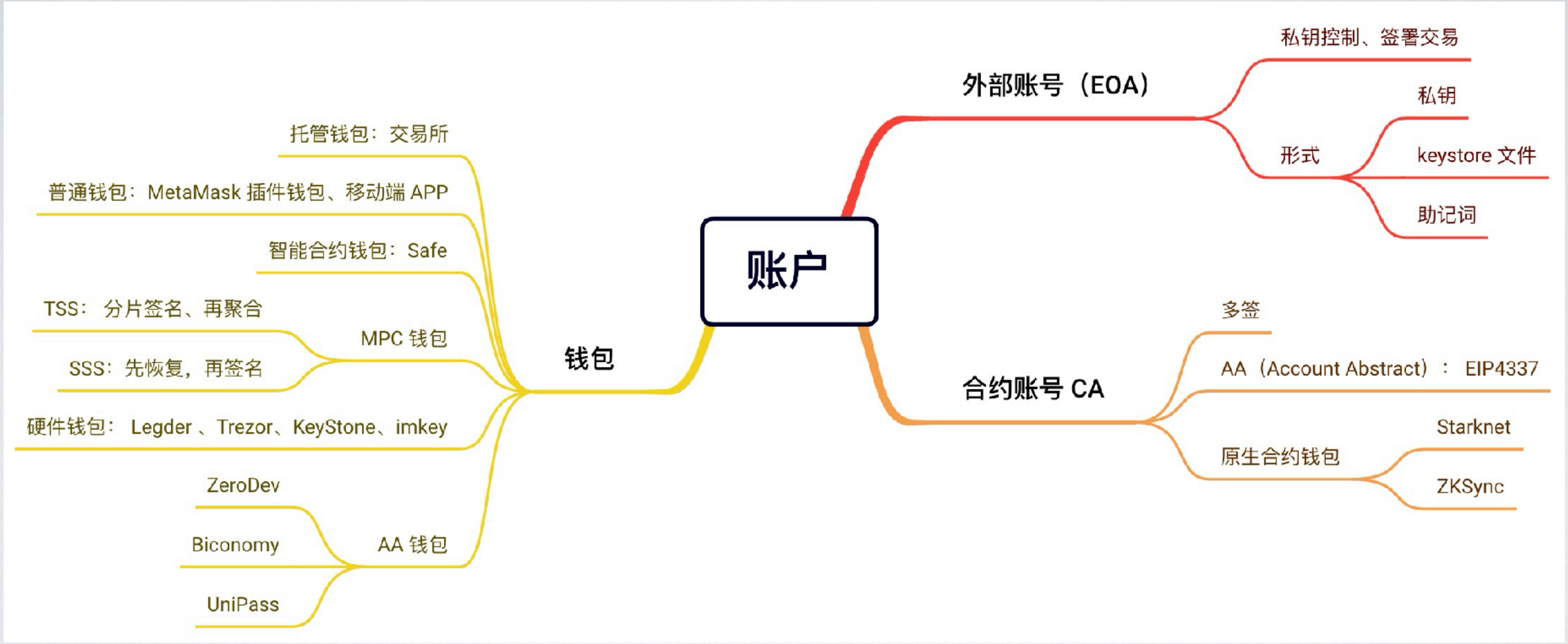
https://learnblockchain.cn/article/5426

智能合约钱包账户

Demo： OpenSpace100/blockchain-tasks/solidity_sample_code/ContractWallet.sol

多签钱包SAFE

https://safe.global/wallet


账户
外部账号（EOA）
私钥控制、签署交易
形式
私钥
keystore 文件
助记词
合约账号 CA
多签
AA（Account Abstract）： EIP4337
原生合约钱包
Starknet
ZKSync
钱包
托管钱包：交易所
普通钱包：MetaMask 插件钱包、移动端 APP
智能合约钱包：Safe
MPC 钱包
TSS： 分片签名、再聚合
SSS：先恢复，再签名
硬件钱包： Legder 、Trezor、KeyStone、imkey
AA 钱包
ZeroDev
Biconomy
UniPass

实践练习

- 使用 Safe 多签：
  - https://decert.me/quests/4d4d50ab-84ab-4289-ac67-e3839e078537
- 实现一个简单的多签钱包， 功能：
  - 多签持有人可提交交易
  - 其他多签人确认交易（使用交易的方式确认即可）
  - 达到多签门槛、任何人都可以执行交易

https://decert.me/quests/f832d7a2-2806-4ad9-8560-a27ad8570c6f