钱包模块：钱包分类

主要有三种类型：轻钱包模式、重钱包模式和兼容模式。

轻钱包模式：

轻钱包模式下，需要有一个开放的节点与钱包通信,可能是Http RPC协议，这个节点可是任意链上的节点。轻钱包通常做成浏览器插件，插件在运行时会自动注入Web3框架，DApp可以通过Web3与区块链节点通信。当DApp只是单纯的获取数据时是不需要钱包介入的，但是当DApp需要发送交易到链上时需要通过钱包完成对交易签名的过程。优点：不需要用户同步区块链节点就可使用

缺点：需要一个公开的节点提供服务，可能会存在安全性问题

重钱包模式：

重钱包会自己同步并持有一个区块链节点，提供一个浏览器环境，其他与钱包相似。优点：自己持有并同步节点，安全性高缺点：需要持有一个全量的区块链节点

兼容模式：

兼容模式可以在轻钱包和重钱包下同时使用，与钱包通信的节点可以选择在钱包外本地持有，也可以自己搭建服务持有并公布节点。

钱包模块：账户类型

以太坊有 2 种账户：外部拥有账户和合约账户：

Externally owned accounts (EOAs)

- An externally controlled account
- has an ether balance,
- can send transactions (ether transfer or trigger contract code),
- is controlled by private keys,
- has no associated code.

Contract accounts

- A contract
- has an ether balance,
- has associated code,
- code execution is triggered by transactions or messages (calls) received from other contracts.when executed - perform operations of arbitrary complexity (Turing completeness) - manipulate its own persistent storage, i.e., can have its own permanent state - can call other contracts

All action on the Ethereum block chain is set in motion by transactions fired from externally owned accounts. Every time a contract account receives a transaction, its code is executed as instructed by the input parameters sent as part of the transaction. The contract code is executed by the Ethereum Virtual Machine on each node participating in the network as part of their verification of new blocks.

钱包模块：账户比较

账户比较：

- 一个外部拥有账户可以通过创建和用自己的私钥来对交易进行签名，来发送消息给另一个外部拥有账户或合约账户。
- 在两个外部拥有账户之间传送的消息只是一个简单的价值转移。
- 从外部拥有账户到合约账户的消息会激活合约账户的代码，允许它执行各种动作。
- 不像外部拥有账户，合约账户不可以自己发起一个交易。相反，合约账户只有在接收到一个交易之后(从一个外部拥有账户或另一个合约账户接)，为了响应此交易而触发一个交易。

钱包模块：钱包安全（备份）

钱包的形态多样，备份的方式多中，根本目的： 防盗，防丢，分散风险。主要有：

多处和分离备份 keystore && password：

- 将 keystore 文件放置多处安全的位置，如离线的 USB 以及你信任的云存储服务商。
- keystone 对应的 password，你应该采用强密码，同样多处且与 keystore 分离备份。

纸钱包备份：

- 纸钱包实质就是将 keystore 或 私钥以纸质化形式保存，一般为二维码形式，直接打印对应的二维码纸钱包。

脑钱包:

- 通过 BIP 39 提案的方式生成足够随机的，可记忆的助记码。

多重签名

- 多重签名是一个不错的选择，它的优势是当你需要提取超过限制的金额时，需要多把私钥同时授权，同时提升防盗，防丢的安全性。

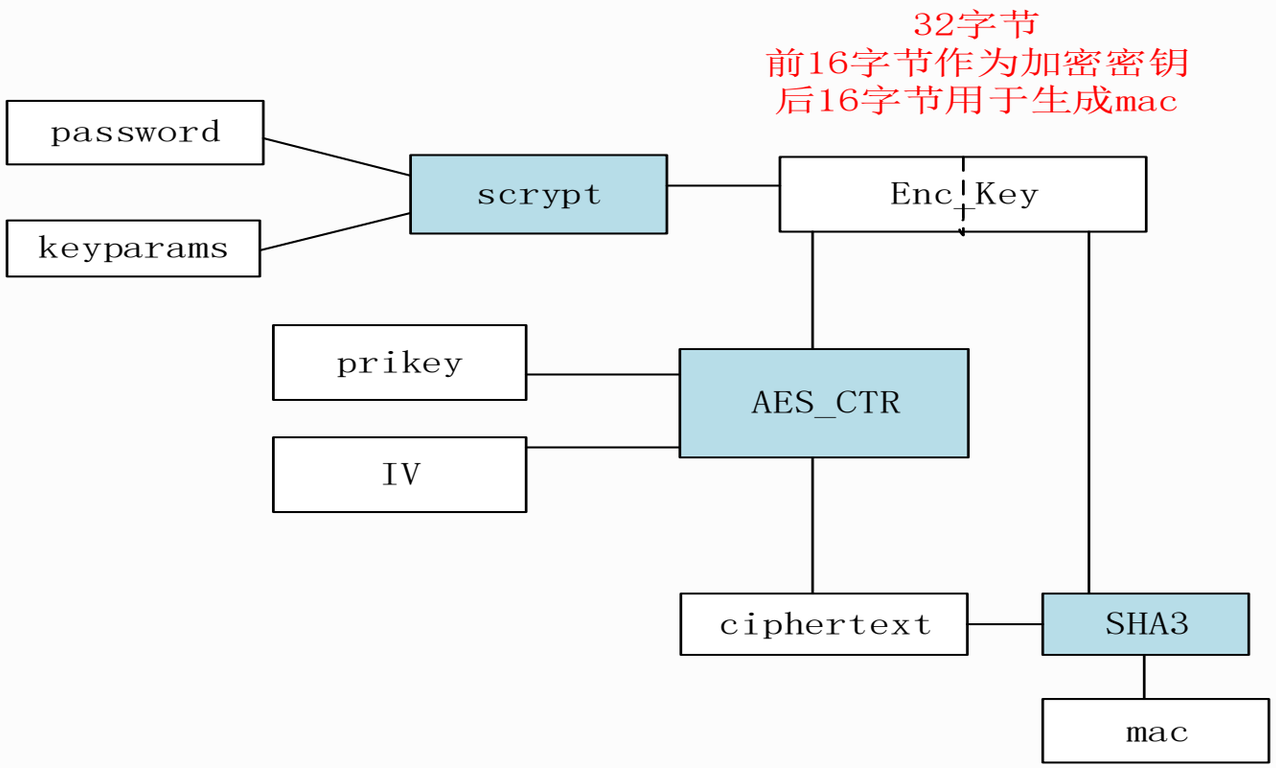
钱包模块：账户文件说明

Keystore目录账户文件格式： 一般 UTC--<created_at UTC ISO8601>-<address hex>，如： UTC--2018-07-02T03-56-34.619039938Z--f8ffd54afc5a41b352af2feec9cc447b90a4cfbb

```
008aeeda4d805471df9b2a5b0f38a0c3bcba786b   地址
{
    "crypto" : {
        "cipher" : "aes-128-ctr",   加密私钥的算法
        "cipherparams" : {          加密需要使用的参数
            "iv" : "83dbcc02d8ccb40e466191a123791e0e"
        },
        "ciphertext" :              私钥加密后的密文
"d172bf743a674da9cdad04534d56926ef8358534d458fffccd4e6ad2fbde479c",
        "kdf" : "scrypt",           密钥生成函数使用的算法
        "kdfparams" : {             scrypt函数使用的参数
            "dklen" : 32,
            "n" : 262144,
            "r" : 1,
            "p" : 8,
            "salt" :
"ab0c7876052600dd703518d6fc3fe8984592145b591fc8fb5c6d43190334ba19"
        },
        "mac" :                     用于校验正确性
"2103ac29920d71da29f15d75b4a16dbe95cfd7ff8faea1056c33131d846e3097"
    },
    "id" : "3198bc9c-6672-5ab3-d995-4942343ae5b6",
    "version" : 3
}
```

- cipher：对称算法的名称，该算法用于加密私钥
- cipherparams：上述 cipher 算法需要的参数;
- ciphertext：以太坊私钥使用上述 cipher 算法进行加密的结果
- kdf：密钥生成函数，用密码生成加密 keystore 文件的密钥
- kdfparams：上述 kdf 算法需要的参数;
- Mac：用于验证密码的正确性。

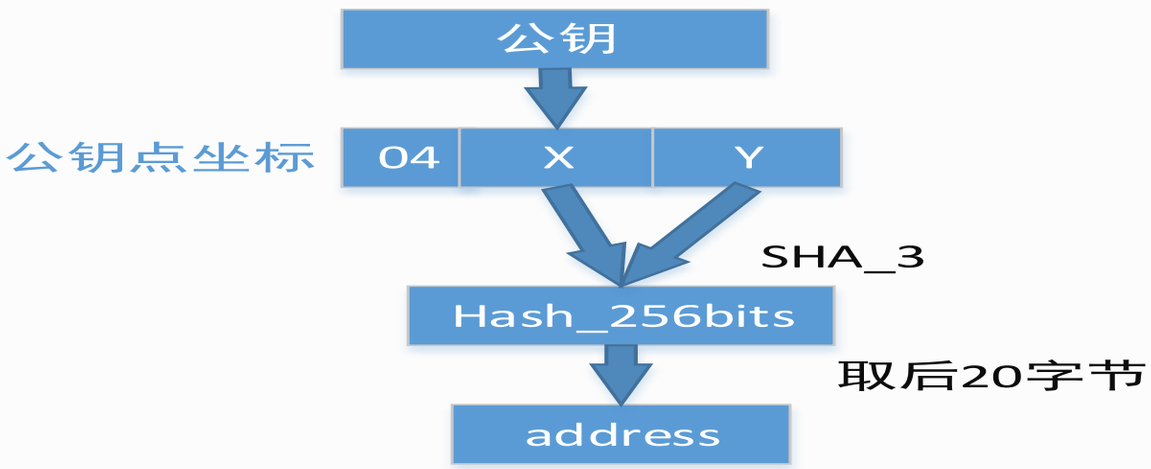
钱包模块：私钥和账户地址生成



① 首先，根据密码和参数keyparams经过scrypt函数来计算加密密钥。

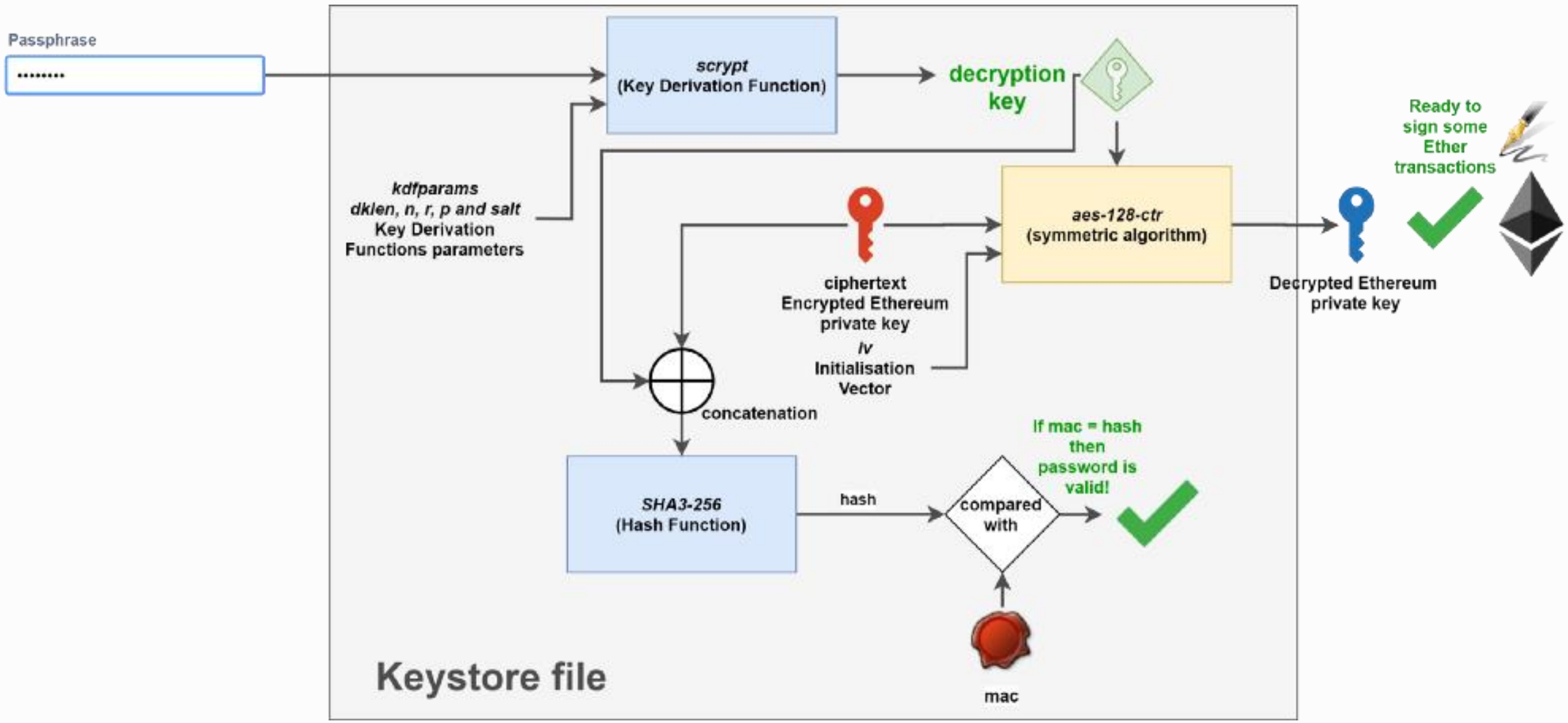
② 然后，用计算出的加密密钥前16字节和参数IV对prikey使用AES_CTR进行加密，得到私钥的密文ciphertext。

③ 最后，将加密密钥后16字节和ciphertext 密文一起用SHA3计算得到mac。



1. 首先，上图中的prikey生成public key。
2. 然后使用Keccak256算法对public key计算hash，取后20字节作为address

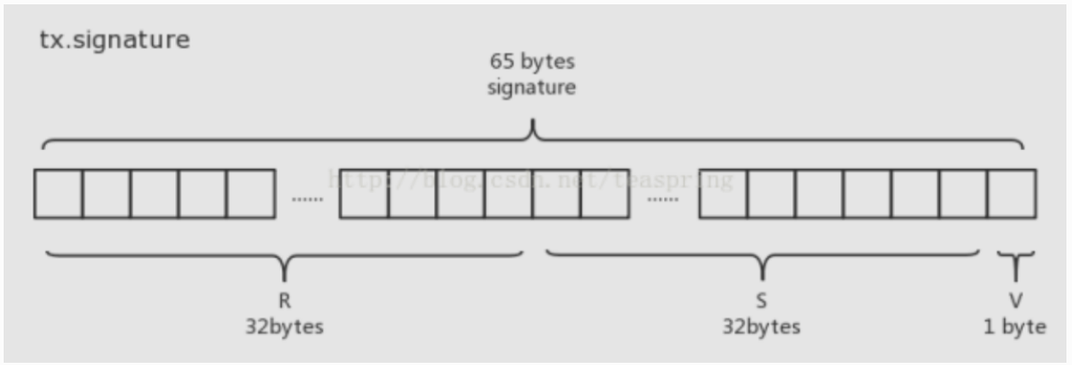
钱包模块：秘钥解密



首先，输入密码，这个密码作为 kdf 密钥生成函数的输入，来计算解密密钥。
然后，计算出的解密密钥和 ciphertext 密文连接做SHA_3，和 mac 比较来确保密码是正确的。
最后，用解密密钥对 ciphertext 密文解密
其实就是读取密钥文件和加密密码，对私钥进行解密，以便使用私钥对发送的交易进行签名

钱包：交易签名和验证

以太坊的交易使用私钥来签名。因此发起交易之前必须要进行对钱包的解锁操作，得到私钥。
签名结果的结构如下：



其中前64字节为签名值。最后一字节为标志位（有效标志位为0,1），用于签名验证时还原公钥。签名算法步骤及标志位的标志方法如下：

- 输入：椭圆曲线参数(q, FR, S, a, b, P, n, h)；私钥d；消息的哈希e=hash(m)
- 输出：签名结果(r, s, v)