**井通公链钱包开发**

目录

[1.钱包简介 2](#_Toc2012669244)

[2.基础功能 4](#_Toc1917060011)

[3.节点配置 10](#_Toc1330448936)

[4.综述 11](#_Toc1255534788)

# 1.钱包简介

钱包的本质就是管理用户的私钥，通过私钥签名交易管理用户在区块链上的数字货币。现在随着加密货币的蓬勃发展，也发展出不同类型的钱包，以适应各种的使用场景。

**按照去中心化程度**，可以把钱包分为全节点钱包、轻钱包、以及中心化钱包。

**全节点钱包**

此种钱包需要先进行软件安装，安装后会与整个区块链进行同步，存储整个区块链，所以全新钱包开始同步时，必须从第一笔数据开始下载，会花上数小时～数十日的时间，并且占用为数不小的存储空间与网络流量。这样的钱包又称为完整节点。由于此种钱包能提供所对应的加密货币网络完整区块链与服务，所以可以提升该加密货币网络的完整性与可靠性，因此某些加密货币，会对持有这种钱包的用户进行奖励

**轻钱包**

又名SPV钱包（simplified payment verification wallet，简单支付验证钱包）

与前者比，此种钱包仅存储私钥与结算，不存储整个区块链，所以占用资源很少，较适用于移动设备。

**中心化钱包**

交易所钱包：各交易所为了能够出入币，也都会提供对应该币的钱包地址。此种钱包不会提供私钥给用户，所以一旦存入，就只能依靠交易所的机制来提领。这同时也算是一种在线钱包。

**按照私钥存储方式，**按照私钥存储方式，可以钱包分为冷钱包、热钱包 。

**冷钱包（Cold Wallet）**

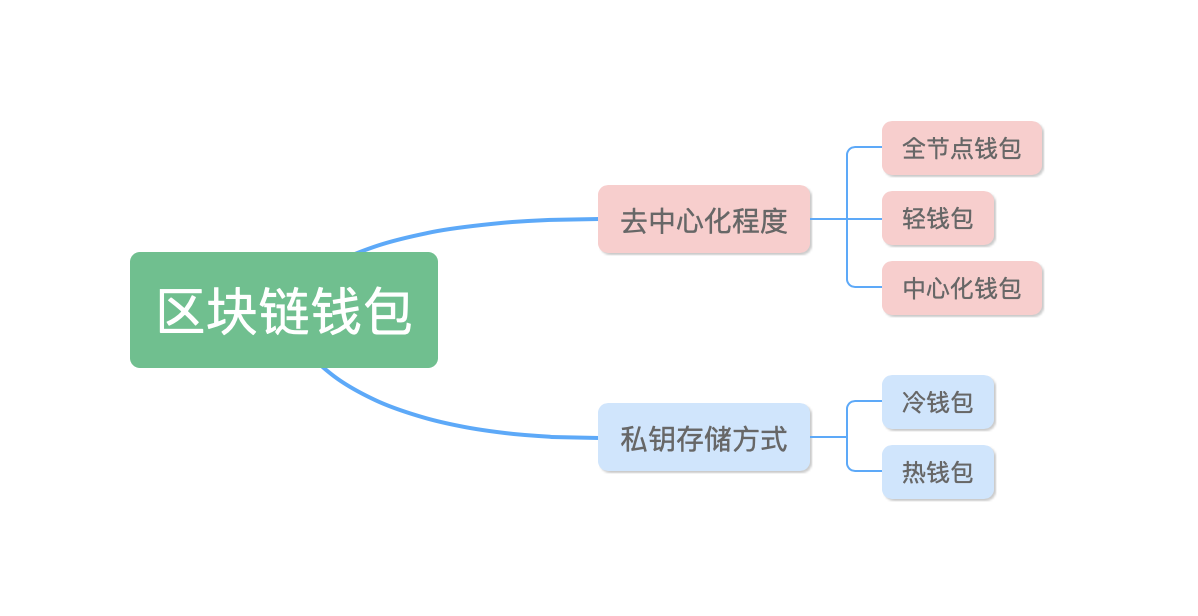
与热钱包相对应，也称离线钱包或者断网钱包，意指网络不能访问到用户私钥的钱包。 冷钱包通常依靠“冷”设备（不联网的计算机、手机等）确保比特币私钥的安全，运用二维码通信让私钥不触网，避免了被黑客盗取私钥的风险，但是也可能面临物理安全风险（比如计算机丢失，损坏等）。

将私钥存储至不会与网络连线的设备上，包括打印、手抄（纸钱包），甚至自行背诵（脑钱包），以确保存储期间的绝对安全。不过上述钱包在进行交易时，仍必须将私钥输入到一般软件钱包，而最高标准的冷钱包，还必须能进行离线签署，只将签署过的交易发出来。

将私钥存储在额外的特制硬件设备，使用时交易需在硬件内部进行交易签署才提交，只要硬件没有被破解，就绝对安全。 如果此硬件设备完全离线、只提交签署过的交易消息，那同时也是最高标准的冷钱包。

**热钱包（Hot Wallet）**

与冷钱包相对应，也称在线钱包或者联网钱包（Online Wallet），也就是网络能够访问到用户私钥的钱包。热钱包因其联网特性，外人可能通过互联网访问用户的私钥，因此安全性比冷钱包低，但比冷钱包更便利。另外，无论是使用冷钱包还是热钱包，私钥都是关键所在。



# 2.基础功能

钱包对外呈现可能有不同的功能，充值、提现、转账等，但从本质上来说只有一个功能，那就是转账。区块链本质上就是一个账本，记录着一笔笔交易，钱包当然离不开这个本质。

在此基础上，钱包一般包含以下基础功能：

1. 创建钱包：生成私钥、keystore、或者生成助记词；

2.备份钱包：导出私钥、keystore、或者助记词；

3.导入钱包：通过keystore、助记词或者私钥导入钱包；

4.转账功能：接收、发送数字资产、扫描二维码进行支付等；

5.查询功能：查询资产余额及交易记录；

6.修改密码等；

**创建钱包**

从技术上和安全上来讲，通过代码直接生成符合私钥规则的、与网络无关的公私钥，私钥通过keystore的形式储在客户端，不直接暴露私钥从而确保私钥的安全。

具体到井通公链，我们可以通过调用jingtum-lib-java来实现，

方法一：Wallet.generate();

方法二：Wallet.fromSecret(secret);

当然也可以通过其它的井通开源库来实现，如jcc\_rpc等。

**keyStore**

keystore就是私钥的再加密形式，它使用用户自定义密码对私钥进行加密。keystore密码是唯一不可修改的，那么钱包密码修改之后，keystore也会相应修改。一定要记住加密keystore的密码，一旦忘记密码，就相当于遗失了该钱包所有的区块链资产

具体可以调用jch-jingtum-lib-android来实现，

Wallet wallet = new Wallet("shExMjiMqza4DdMaSg3ra9vxWPZsQ");

try {

KeyStoreFile keyStoreFile = KeyStore.createLight("Key123456", jtKeyPair);

System.out.println(keyStoreFile.toString());

} catch (CipherException e) {

e.printStackTrace();

}

生成的keystore格式是JSON字符串：

{

"address": "jHY6aRcs7J6KnfgqD4FVwTQ247boj9nbDZ",

"id": "1c1bf720-82fd-4ed3-bddf-72ebbc7b4262",

"version": 3,

"crypto": {

"cipher": "aes-128-ctr",

"ciphertext": "0bc63928ace81eb82869d5008372830191bad7706ef2101665d009a9e6",

"cipherparams": {

"iv": "2ae846f498bbb6ff6a7d572d51cdd74b"

},

"kdf": "scrypt",

"kdfparams": {

"dklen": 32,

"n": 4096,

"p": 6,

"r": 8,

"salt": "944611340b628e66850eff427ec0df006788d2aa7e3809b383dbe05282edd723"

},

"mac": "ad1343750c048c96b019dc09dd6a5b93d5664cfd5147dd052ec040546d53617f"

}

}

***需要强调的是，为了保证keystore的加密强度，建议密码需要一定的复杂度，比如大小写，长度限制等。***

**转账**

***Gas Limit***称为限额，它是用户愿意在一笔交易中花费Gas的最大数量。交易所需的Gas是通过调用智能合约执行多少代码来定义。 如果不想花太多的Gas，通过降低Gas Limit将不会有太大的帮助。 因为必须包括足够的Gas来支付的计算资源，否则由于Gas不够导致交易失败，设置的所有Gas limit也将消耗光。建议在有充足SWTC情况下，将Gas Limit尽量设高，所有未使用的Gas将在转账结束时退还。

通过降低***Gas Price***可以节省矿工费用，但是也会减慢矿工打包的速度。矿工会优先打包 Gas Price设置高的交易，如果想加快转账，可以把Gas Price设置得更高，这样就可以排队靠前。如果不急，只需要设置一个安全的 Gas Price，矿工也会打包。

***所以在开发钱包时，最好可以提供让用户手动调整Gas Price的选项。***

依旧通过调用jingtum-lib-java来实现，

// 测试环境

String server = "ws://ts5.jingtum.com:5020";

// 是否使用本地签名方式提交交易

Boolean local\_sign = true;

Connection conn = ConnectionFactory.getCollection(server);

Remote remote = new Remote(conn, local\_sign);

// 转账地址

String account = "j3UcBBbes7HFgmTLmGkEQQShM2jdHbdGAe";

// 钱包秘钥

String secret = "ssWiEpky7Bgj5GFrexxpKexYkeuUv";

// 收款地址

String to = "jNn89aY84G23onFXupUd7bkMode6aKYMt8";

// 转账数量

String value = "0.01";

// 手续费

String fee = "100000";

// token名称

String token = "";

// 发行方地址

String issuer = "";

AmountInfo amount = new AmountInfo();

amount.setCurrency(token);

amount.setValue(value);

amount.setIssuer(issuer);

Transaction tx = remote.buildPaymentTx(account, to, amount);

tx.setSecret(secret);

// 添加交易备注

List<String> memos = new ArrayList<String>();

memos.add("SWT转账");

tx.addMemo(memos);

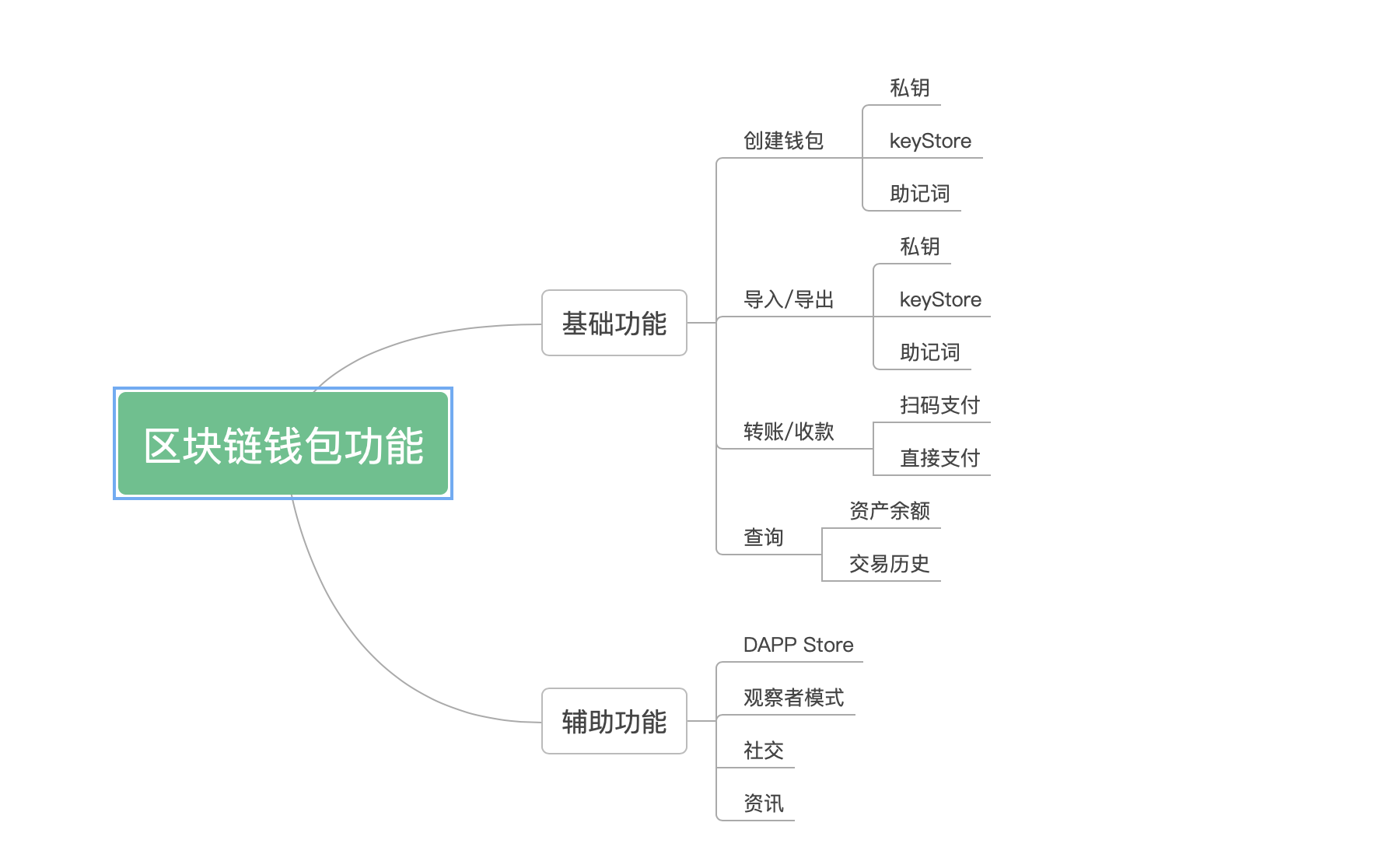
tx.setFee(fee);

TransactionInfo bean = tx.submit();

同样，我们也可以通过其它的井通开源库来实现。

除了上述的基础功能外，钱包APP也可以有其他辅助功能：

1. DAPP Store
2. 观察地址（只导入钱包地址以监听某些钱包地址的token数量）；
3. 社交；
4. 资讯信息（交易类、项目方/币方信息、行业信息等）；



# 3.节点配置

由于每个节点同步到的数据进度差距太大，没办法像中心化的业务一样做负载均衡，只能通过热备的形式来保证当一个节点出现故障时能够快速切换到另外一个节点。因为区块打包本身就比较耗时，因此此处的时效性要求还是可以容忍的。这里以jingtum-lib-java为例，基于ws协议跟底层交互，其中ws封装到Server类中，Server类是一个内部类，不对外开放；Server类封装在Remote类中，Remote类提供对外访问接口并可创建两类对象：Get方式请求的Request对象和Post方式请求的Transaction对象，这两类对象都通过submit()方法提交数据到底层，直接和节点交互。我们可以通过判断Remote链接节点失败时主动切换节点地址。<http://www.swtcdocs.org/zh_CN/latest/wiki/node/> 这里提供井通公共节点地址，可以通过WS://方式访问。

# 4.综述

上述的内容更多的时偏向于去中心化区块链钱包的开发，当然整体的框架结构基本都是一样的。

