

**PFP14井通区块链学习大纲 JINGTUMLIB**

Author:蒲苇

目录

[一、 jngtum-lib是个啥玩意儿](#_Toc20111_WPSOffice_Level1) [3](#_Toc20111_WPSOffice_Level1)

[二、 先安装](#_Toc14705_WPSOffice_Level1) [3](#_Toc14705_WPSOffice_Level1)

[三、 看看链上的区块（账本）信息](#_Toc7331_WPSOffice_Level1) [4](#_Toc7331_WPSOffice_Level1)

[四、 搞个钱包玩玩](#_Toc11184_WPSOffice_Level1) [6](#_Toc11184_WPSOffice_Level1)

[五、 激活转账](#_Toc2321_WPSOffice_Level1) [7](#_Toc2321_WPSOffice_Level1)

[六、 通证的转账](#_Toc29699_WPSOffice_Level1) [10](#_Toc29699_WPSOffice_Level1)

[七、 挂个单](#_Toc476_WPSOffice_Level1) [11](#_Toc476_WPSOffice_Level1)

[八、 查交易](#_Toc9329_WPSOffice_Level1) [15](#_Toc9329_WPSOffice_Level1)

[九、 智能合约](#_Toc28614_WPSOffice_Level1) [18](#_Toc28614_WPSOffice_Level1)

[十、 本地签名和不签名的玩法](#_Toc28222_WPSOffice_Level1) [18](#_Toc28222_WPSOffice_Level1)

1. jngtum-lib是个啥玩意儿

井通区块链是基于区块链技术构建的一个去中心化互享生态的互联网交易网络，简单点说，就是区块链技术+交易网络。作为开发者，我们怎么跟链互动，开发出自己的产品？jingtum-lib就是那条最直接的交互通道。

三方面认识jingtum-lib：

1.基于websocket的底层节点公共库；

2.两种请求方式：get请求和post请求。其中get请求组装成Request对象，post请求组装成Transaction对象，两种对象都通过submit方法提交数据，从而跟底层交互；

3.前后台都适用（gulp自动编译成前端js库）。

1. 先安装

1.作为后端库安装

npm install --save jingtum-lib

2.作为前端库安装

gulp自动编译成前端js库，存放于dist/jingtum-lib-xxx.js和dist/jingtum-lib-xxx.min.js, 根据需求直接引入即可。

<script src="目录/jingtum-lib-2.2.0.min.js"></script>

3.源码

https://github.com/jingtum/jingtum-lib

1. 看看链上的区块（账本）信息

a.先看下最新账本跑到哪儿了

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://xxx:port', local\_sign:true});

remote.connect(function (err, result) {  
 if (err) {  
 console.log('err:', err);  
 } else {  
 var req = remote.requestLedgerClosed();  
 req.submit(function (err, result) {  
 if (err) {  
 console.log('err:', err);  
 }  
 else {  
 console.log(result);  
 }  
 });  
 }  
});

返回结果：

{

ledger\_hash: 'C782A51A3405484B99DD1E0F5962D6D8629845240E973F47D72F6283BB502BF3',

ledger\_index: 3594119

}

返回结果说明：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| ledger\_hash | String | 账本hash |
| ledger\_index | Number | 账本高度/区块高度 |

b.具体某个账本里有啥信息

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://xxx:port', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 return console.log('err:',err);  
 }

//var req = remote.requestLedger({});//将默认返回最新账本信息  
 var req = remote.requestLedger({  
 ledger\_index: 3637979,   
 transactions: true   
 });  
 req.submit(function(err, result) {  
 if(err) {console.log('err:',err);}  
 else if(result){  
 console.log(result);  
 }  
 });  
});

返回结果：

{

accepted: true,

account\_hash: '81A32C91AB405F1761A98088A92F307D40A1F77DEF43DD25DFD2E619B3DF390E',

close\_time: 615463820,

close\_time\_human: '2019-Jul-03 10:10:20',

close\_time\_resolution: 10,

closed: true,

hash: 'FBE16D38BE8B8FAAC139D8162183D6DA4D0BC88DCD267644ADC4A95679458CCB',

ledger\_hash: 'FBE16D38BE8B8FAAC139D8162183D6DA4D0BC88DCD267644ADC4A95679458CCB',

ledger\_index: '3637979',

parent\_hash: 'B0EE6597B5F0AD98A13C83AE7F1EAF1AECE7CAD10E0586E7EED9B7230A00E46DD',

seqNum: '3637979',

totalCoins: '600000000000000000',

total\_coins: '600000000000000000',

transaction\_hash: '05B107D3F0041A23C0D758A39290ED01B3E4BD686B7B978BF78C9461A9117A26',

transactions:

[ 'C79B437A27DEF0761F98421190D4FFDB2F4F9947C7089E6E9AF04D9BF3FCD70A' ]

}

返回结果说明：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| accepted | Boolean | 区块是否已经产生 |
| account\_hash | String | 状态hash树根 |
| close\_time | Integer | 关闭时间 |
| close\_time\_human | String | 关闭时间 |
| close\_time\_resolution | Integer | 关闭周期 |
| closed | Boolean | 账本是否已经关闭 |
| hash | String | 账本hash |
| ledger\_hash | String | 账本hash |
| ledger\_index | String | 账本高度/区块高度 |
| parent\_hash | String | 上一区块hash值 |
| seqNum | String | 账本高度/区块高度 |
| totalCoins | String | swt总量 |
| total\_coins | String | swt总量 |
| transaction\_hash | String | 交易hash树根 |
| transactions | Array | 该账本里的交易列表 |

1. 搞个钱包玩玩

**两种方式创建钱包**

var jlib = require('jingtum-lib');

**1.无私钥，随机生成**

var w1 = jlib.Wallet.generate();

console.log('w1: ', w1);

**2.有私钥，导出钱包**

var w2 = jlib.Wallet.fromSecret('sn37nYrQ6KPJvTFmaBYokS3FjXUWd');

console.log('w2: ', w2);

**3.知识扩展**

文档《井通加密算法v2》详细介绍以下三点：

a.井通钱包地址生成流程；

b.公钥、私钥、钱包地址的关系；

c.井通加密算法；

1. 激活转账

**概念普及**

激活：钱包生成之后并没有上链，类似银行开户，井通系统一般需要30个swtc将钱包激活上链。

节点：整个井通区块链底层是由n台服务器管理存储信息，并且相互验证。这些服务器就是节点。

燃料：每执行一次交易会消耗固定的燃料，当前每笔0.01swtc。

hash：通过哈希算法生成的一串数字和字母组成的散列字符串，一般作为事务的唯一标识值。

**具体实现**

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020',

local\_sign:true});

remote.connect(function(err, result) {

if (err) {

return console.log('err:',err);

}

var tx = remote.buildPaymentTx({

account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

to: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',

amount: {

"value": 30,

"currency": "SWT",

"issuer": ""

}

});

tx.setSecret('sn37nYrQ6KPJvTFmaBYokS3FjXUWd');

tx.addMemo('支付30swtc.');//可选

tx.submit(function(err, result) {

if(err) {console.log('err:',err);}

else if(result){

console.log('res:', result);}

});

});

返回结果：

{ engine\_result: 'tesSUCCESS',

engine\_result\_code: 0,

engine\_result\_message: 'The transaction was applied. Only final in a validated ledger.',

tx\_blob: '120000220000000024000000982F24EE8B62614000000001C9C380684000000000002710732102FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD16774463044022059155C39AB14477858F7F57CF59D6EDEE58749A711822029360A497D13B7B2E3022005964684DFDB21CCB70708A84917B3BE4816C9A4DFF67460822FF6D3188B0A37811472F05993EBA9858291D364EBF6EEC3D851BD3792831485B6C98BAD6DBF7805D3C5CCC1B4F989E0CE6749F9EA7D0DE694AFE4BB983330737774632EE1F1',

tx\_json:

{ Account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

Amount: '30000000',

Destination: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',

Fee: '10000',

Flags: 0,

Memos: [ [Object] ],

Sequence: 152,

SigningPubKey: '02FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD167',

Timestamp: 619613026,

TransactionType: 'Payment',

TxnSignature: '3044022059155C39AB14477858F7F57CF59D6EDEE58749A711822029360A497D13B7B2E3022005964684DFDB21CCB70708A84917B3BE4816C9A4DFF67460822FF6D3188B0A37',

hash: '6CC96A8C2B47AD9F3837C576F43DBCC188432C1101DC3387906D83DD327D663F' } }

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| engine\_result | | String | 请求结果 |
| engine\_result\_code | | Array | 请求结果编码 |
| engine\_result\_message | | String | 请求结果message信息 |
| tx\_blob | | String | 16进制签名后的交易 |
| tx\_json | | Object | 交易内容 |
|  | Account | String | 账号地址 |
| Amount | String | 交易金额 |
| Destination | String | 对家 |
| Fee | String | 交易费 |
| Flags | Integer | 交易标记 |
| Memos | Array | 备注 |
| Sequence | Integer | 单子序列号 |
| SigningPubKey | Object | 签名公钥 |
| TransactionType | String | 交易类型 |
| TxnSignature | String | 交易签名 |
| hash | String | 交易hash |

1. 通证的转账

通证转账同激活转账一样，不同的是参数中币种与币种发行方

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://xxx:port', local\_sign:true});

remote.connect(function(err, result) {

if (err) {

return console.log('err:',err);

}

var tx = remote.buildPaymentTx({

account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

to: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',

amount: {

"value": 10,

"currency": "CNY",

"issuer": "jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS"

}

});

tx.setSecret('sn37nYrQ6KPJvTFmaBYokS3FjXUWd');

tx.addMemo('支付10CNY.');//可选

tx.submit(function(err, result) {

if(err) {console.log('err:',err);}

else if(result){

console.log('res:', result);

}

});

});

1. 挂个单

1.创建挂单

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://xxx:port', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var options = {

type: 'Sell', //买卖类型

account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ', //挂单账号

taker\_gets: {value: '1', currency: 'CNY',issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS' },//对方获得

taker\_pays: { value: '200', currency: 'VCC', issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS'},//对方支付

platform: 'jB77p5n7ekApSzbefoHpuqopPX3k49u71i'

};

var tx = remote.buildOfferCreateTx(options);

tx.setSecret('sn37nYrQ6KPJvTFmaBYokS3FjXUWd');

tx.submit(function (err, result) {

if (err) {

console.log('err:', err);

}

else if (result) {

console.log('res:', result);

}

});

});

返回结果：

result1: { engine\_result: 'tesSUCCESS',

engine\_result\_code: 0,

engine\_result\_message: 'The transaction was applied. Only final in a validated ledger.',

tx\_blob

tx\_json:

{ Account: 'jpX8SEpM387c9tpdAUfBr2gYTfC2k7RatA',

Fee: '10000',

Flags: 0,

Platform: 'jB77p5n7ekApSzbefoHpuqopPX3k49u71i',

Sequence: 3,

SigningPubKey: '02E10087455D301201958DD926D2D195999E88DB1424A3C2592DE058C8732D56F9',

TakerGets:

{ currency: 'CNY',

issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',

value: '200' },

TakerPays:

{ currency: 'VCC',

issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',

value: '200' },

TransactionType: 'OfferCreate',

TxnSignature: '30450221008BC0D2B07BA7AE2424E1A51E86B74357752C812A5A7F4924E316139E28E660A702203F5D95BE98336D2708FED1DB0840379B3AD0CD37AD6B039C4E209BA0897275CB',

hash: 'F3A8B0A58263E1C9992192126EC11CB0D103F6AAE6E8E09BD490DC89177DB47E' } }

返回结果说明：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | | | 类型 | 说明 |
| engine\_result | | | String | 请求结果 |
| engine\_result\_code | | | Array | 请求结果编码 |
| engine\_result\_message | | | String | 请求结果message信息 |
| tx\_blob | | | String | 16进制签名后的交易 |
| tx\_json | | | Object | 交易内容 |
|  | Account | | String | 账号地址 |
| Fee | | String | 交易费 |
| Platform | | String | 平台标记账号，设置了才有此字段 |
| Flags | | Integer | 交易标记 |
| Sequence | | Integer | 单子序列号 |
| SigningPubKey | | String | 签名公钥 |
| TakerGets | | Object | 对家得到的 |
|  | currency | String | 货币 |
| issuer | String | 货币发行方 |
| value | String | 额度 |
| TakerPays | | String | 对家支付的； |
| Timestamp | | Integer | 时间戳，签名方式没有该字段 |
| TransactionType | | String | 交易类型：TrustSet信任;RelationDel解冻；RelationSet授权/冻结 |
| TxnSignature | | String | 交易签名 |
| hash | | String | 交易hash |

2.取消挂单

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://xxx:port', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var options = {

account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ', //挂单账号

sequence: 133//要取消的单子号

};

var tx = remote.buildOfferCancelTx(options);

tx.setSecret('sn37nYrQ6KPJvTFmaBYokS3FjXUWd');

tx.submit(function (err, result) {

if (err) {

console.log('err:', err);

}

else if (result) {

console.log('res:', result);

}

});

});

返回结果：

result1: { engine\_result: 'tesSUCCESS',

engine\_result\_code: 0,

engine\_result\_message: 'The transaction was applied. Only final in a validated ledger.',

tx\_blob: '1200082200000000240000009B2F25022E36201900000085684000000000002710732102FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD1677446304402205854DB0E7B53F4348F8899765C4DBB3B9EA399C904FEB9FEE5AC012449BE29CE0220648158D4FA8D1C18E2CF219F83D71985E743618B5082E0586068E8E8AE3C2243811472F05993EBA9858291D364EBF6EEC3D851BD3792',

tx\_json:

{ Account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

Fee: '10000',

Flags: 0,

OfferSequence: 133,

Sequence: 155,

SigningPubKey: '02FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD167',

TransactionType: 'OfferCancel',

TxnSignature: '304402205854DB0E7B53F4348F8899765C4DBB3B9EA399C904FEB9FEE5AC012449BE29CE0220648158D4FA8D1C18E2CF219F83D71985E743618B5082E0586068E8E8AE3C2243',

hash: '4ECC45C9E86C8B8654469F9223C11CAEB944EEB8405BDAD330EA00D33D8EE4F1' } }

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| engine\_result | | String | 请求结果 |
| engine\_result\_code | | Array | 请求结果编码 |
| engine\_result\_message | | String | 请求结果message信息 |
| tx\_blob | | String | 16进制签名后的交易 |
| tx\_json | | Object | 交易内容 |
|  | Account | String | 账号地址 |
| Fee | String | 交易费 |
| Flags | Integer | 交易标记 |
| OfferSequence | Integer | 取消的单子号 |
| Sequence | Integer | 单子序列号 |
| SigningPubKey | String | 签名公钥 |
| Timestamp | Integer | 时间戳，签名方式没有该字段 |
| TransactionType | String | 交易类型：OfferCancel取消订单 |
| TxnSignature | String | 交易签名 |
| hash | String | 交易hash |
| Account | String | 账号地址 |
| Fee | String | 交易费 |
| Flags | Integer | 交易标记 |
| OfferSequence | Integer | 取消的单子号 |
| Sequence | Integer | 单子序列号 |

1. 查交易

交易hash是交易的唯一标识，可通过hash查询某一具体信息。

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://xxx:port', local\_sign:true});

remote.connect(function(err, result) {

if (err) {

return console.log('err:',err);

}

var req = remote.requestTx({

hash:'744A689B030E2F6F1CBFF94B0E52C0F1CDEED5B85D86ACAA0BC76F42C16A2AFC'

});

req.submit(function(err, result) {

if(err) {console.log('err:',err);}

else if(result){

console.log('res:', result);

}

});

});

返回结果：

{

Account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

Amount: '1000000',

Destination: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',

Fee: '10000',

Flags: 0,

Memos: [ { Memo: [Object] } ],

Sequence: 32,

SigningPubKey: '02FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD167',

Timestamp: 615464508,

TransactionType: 'Payment',

TxnSignature: '3044022009294437FD19BC2574C2A193CF72577CA2E0CE09B56A0C506EA1C35E3ECCAAE902204929895243312401161767B22504CFE07D5974BEED79FE8DCC1C121C5865BD03',

date: 615464520,

hash: '744A689B030E2F6F1CBFF94B0E52C0F1CDEED5B85D86ACAA0BC76F42C16A2AFC',

inLedger: 3638049,

ledger\_index: 3638049,

meta:

{ AffectedNodes: [ [Object], [Object], [Object] ],

TransactionIndex: 0,

TransactionResult: 'tesSUCCESS',

delivered\_amount: 'unavailable' },

validated: true

}

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| Account | | String | 钱包地址 |
| Platform | | String | 平台标识账号（挂单中设置了platform时，此参数才有） |
| Amount | | String/Object | 交易金额 |
| Destination | | String | 交易对家地址 |
| Fee | | String | 交易费 |
| Flags | | Integer | 交易标记 |
| Memos | | Array | 备注 |
| Sequence | | Integer | 自身账号的交易号 |
| SigningPubKey | | String | 签名公钥 |
| Timestamp | | Integer | 交易提交时间戳 |
| TransactionType | | String | 交易类型 |
| TxnSignature | | String | 交易签名 |
| date | | Integer | 交易进账本时间 |
| hash | | String | 交易hash |
| inLedger | | Integer | 交易所在的账本号 |
| ledger\_index | | Integer | 账本高度 |
| meta | | Object | 交易影响的节点 |
|  | AffectedNodes | Array | 受影响的节点 |
| TransactionIndex | Integer | -- |
| TransactionResult | String | 交易结果 |
| delivered\_amount | String | -- |
| validated | | Boolean | 交易是否通过验证 |

1. 智能合约

**智能合约是啥**

智能合约是一套以数字形式定义的承诺，承诺控制着数字资产并包含了合约参与者约定的权利和义务，由计算机系统自动执行。

智能合约程序不只是一个可以自动执行的计算机程序，它本身就是一个系统参与者，对接收到的信息进行回应，可以接收和储存价值，也可以向外发送信息和价值。这个程序就像一个可以被信任的人，可以临时保管资产，总是按照事先的规则执行操作。

**井通智能合约现状**

a.采用solidity实现；

b.支持ERC20和ERC721；

c.编译器版本目前支持到5.4；

智能合约还没有上正式环境，有兴趣的小伙伴可以先玩玩

公测说明：https://github.com/jingtum/jingtum-contract-turtoial

1. 本地签名和不签名的玩法

**签名是啥**

在现实社会中，签名作为签名者身份的一种证明，签名代表对签名文件的认可，不可抵赖。在区块链交易中，签名就是只有账号所有者才能生成的一段防伪造的字符串。通过验证该字符串一方面证明交易是本人发起的，另一反面证明交易信息在传输过程中没有被篡改。

**为啥签名**

防篡改：通过对签名的验证，可以保证信息在传输过程中未被篡改。

验证数据的完整性：与防篡改同理，如果信息发生丢失，签名将不完整，解开数字签名和之前的比较就会出现不一致，因而可保证文件的完整。

保密性：对于全级别要求较高的数据，数字签名加密后传输，保证数据在被中途截取后无法获得其真实内容；有利于保证数据的安全性。

防重放：在数字签名中，如果采用了对签名报文添加流水号、时戳等技术，可以有效防止重放攻击。

**怎么签名**

本地签名不需要使用者知道具体流程，所涉及到的序列化及签名过程，lib自动封装。使用者只需通过local\_sign属性来决定是否签名，local\_sign为true时，自动以签名形式跟底层交互，默认false。表现形式为提交给底层的数据是否有明文secret字段。是否签名只对post类请求（需提供私钥）有影响。

**以取消挂单为例**

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://xxx:port', **local\_sign:true**});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var options = {

account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ', //挂单账号

sequence: 133//要取消的单子号

};

var tx = remote.buildOfferCancelTx(options);

tx.setSecret('sn37nYrQ6KPJvTFmaBYokS3FjXUWd');

**tx.setSequence(6680);//可选，签名方式可设置sequence属性。若没有设置，lib自动补全，建议自己设置。**

tx.submit(function (err, result) {

if (err) {

console.log('err:', err);

} else if (result) {

console.log('res:', result);

}

});

});

签名形式可自己设置sequence属性。若没有设置，lib自动补全，建议自己设置。

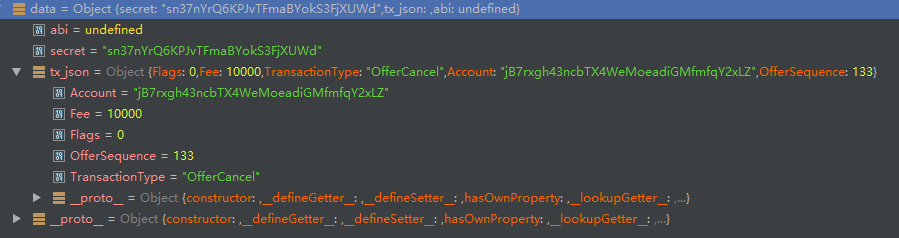
sequence来源：

请求账号信息接口，即remote.requestAccountInfo({account: address, type: 'trust'})，

返回结果中包含Sequence字段。

**请求数据比较**

我们拿前面取消挂单的例子为例，当local\_sign为false时，你传给底层的数据是这样的，如下图。secret以明文的形式传给底层，安全是隐患。



而当我们把local\_sign为true时，你传给底层的数据变成如下图所示的格式。secret你看不到了，只能看到一个整体序列化且加密的tx\_blob字符串，即使捕获截取到该串，也很难破获其明文，相对更安全。

