**以太Proof中Key的处理分析：**

**现象：**

先看一个查询

proof req is:

{  
    **"jsonrpc"**:**"2.0"**,  
    **"method"**:**"eth\_getProof"**,  
    **"params"**:[  
        **"0x10BCC4B6C2555fF48540571ebe5aBa6D32915250"**, //合约地址  
        [  
            **"0x02d9f8353bca53bc9b195aa186ab6d98b49a9120c00257ee2c7d860c26f864ea" // store的key**  
        ],  
        **"0x79ad94" // 查询的高度**  
    ],  
    **"id"**:**1**  
}

proof rsp is:

{  
    **"jsonrpc"**:**"2.0"**,  
    **"result"**:{  
        **"accountProof"**:[  
            **""**,  
            **""**,  
            **""**,  
            **""**,  
            **""**,  
            **""**,  
            **"0xf87180a08fbdd95c5250c6d309bf062d1718aa67e07e2b8959c38915865930facd5e515a8080808080808080a00f867817b68ba58735bcf2635efb8a25b41d79a5ccd6bafb8c76bd20f4b9d3c88080a05d16ec4ebca0c90149fb623fa8d5d0b8340e02cba62f0800b75c445e0d1c0775808080"**,  
            **"0xf8669d311010770374ee72a5655c343d4c3fc9914ddbc79b4f4647dd8257d668b846f8440180a070efb9dc368254b0645df7d904d0e3318cadd9fa9c154677a0c968fce9c63d9fa004c3063bdff2748cfc1e9690ebe993875894b4843918014099f30abe04ffe65d"**  
        ],  
        **"address"**:**"0x10bcc4b6c2555ff48540571ebe5aba6d32915250"**,  
        **"balance"**:**"0x0"**,  
        **"codeHash"**:**"0x04c3063bdff2748cfc1e9690ebe993875894b4843918014099f30abe04ffe65d"**,  
        **"nonce"**:**"0x1"**,  
        **"storageHash"**:**"0x70efb9dc368254b0645df7d904d0e3318cadd9fa9c154677a0c968fce9c63d9f"**,  
        **"storageProof"**:[  
            {  
                **"key"**:**"0x2d9f8353bca53bc9b195aa186ab6d98b49a9120c00257ee2c7d860c26f864ea"**,  
                **"proof"**:[  
                    **""**,  
                    **""**,  
                    **"0xf8d1a04b04eb1bb042990ec098db0d02f51062bc76d71b26ec1789253a7c70329e2f9980a0816499520f593d56ae99bdba3eb3ae1651a9c75298083a0aa2b99aa68a20ec4380808080a0a77d00d3a762bf0ed1e7ae78fe8b1e2a8b09d956e3e25bd3209d1af4d0a6252fa00bbd63bfa772ca4b9f7883ca83abdb7595dd40f868482b3b02446f1f9fb377f6808080a01f83049ba2377c55daa3445a11998a88802357079ef7c25f82e95dacee50456a80a0f2109e9a0e37462aaa074eacdb16c37bc04114b41aa67541cce065ed8d74acee8080"**,  
                    **"0xf85180808080a0f1db927de76f2a5d7a019999a5bfa62557502a9c4aa0a2d9a55d0186e483ae6e808080a073fbe663f4fe64f58f96db116df67e3115ab3813c1c61026e58fef797bd183c38080808080808080"**,  
                    **"0xf8429f2040a2ede5d9b19fd502c4f2105d43540bbc6f91660d7fcbc47963f8db6939a1a06f754faf1fc54b6b6204402972a1066335a37bb89f817f86ec34d4fe8991b2c3"**  
                ],  
                **"value"**:**"0x6f754faf1fc54b6b6204402972a1066335a37bb89f817f86ec34d4fe8991b2c3"**  
            }  
        ]  
    },  
    **"id"**:**1**  
}

可以看到查询proof的指定的key为

0x02d9f8353bca53bc9b195aa186ab6d98b49a9120c00257ee2c7d860c26f864ea

而返回的proof的结果中key为

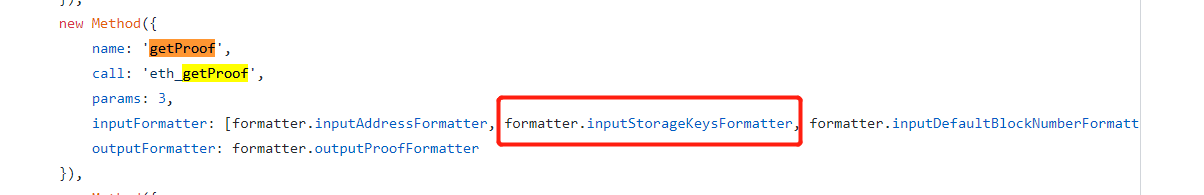
0x2d9f8353bca53bc9b195aa186ab6d98b49a9120c00257ee2c7d860c26f864ea

**结果：**

去掉了最前端的0

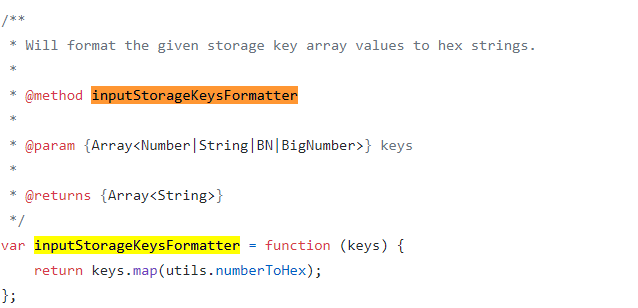
**原因：**

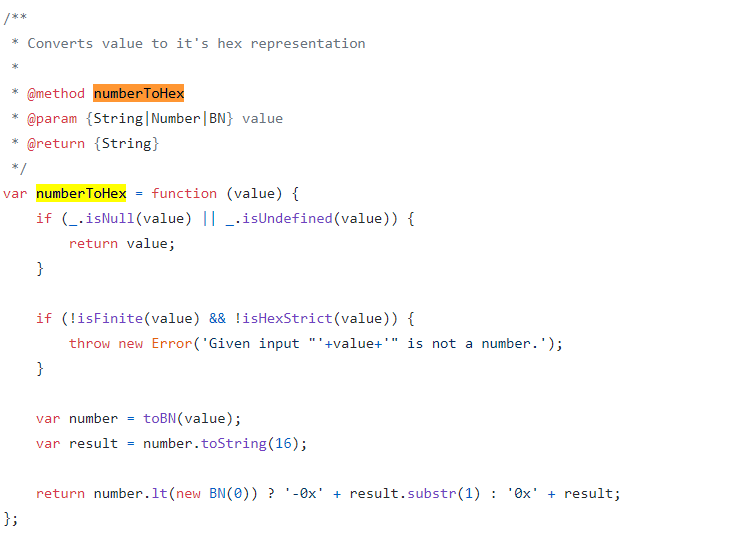
1. Rpc参数格式化

****

Getproof的RPC参数首先使用formatter.inputStorageKeysFormatter进行格式化

1. 格式化处理





主要处理过程：

var number = toBN(value);

var result = number.toString(16);

return number.lt(new BN(0)) ? '-0x' + result.substr(1) : '0x' + result;

将字符串的key转为big number，然后转为16进制的hex string，如果是正数，则最前端加0x。

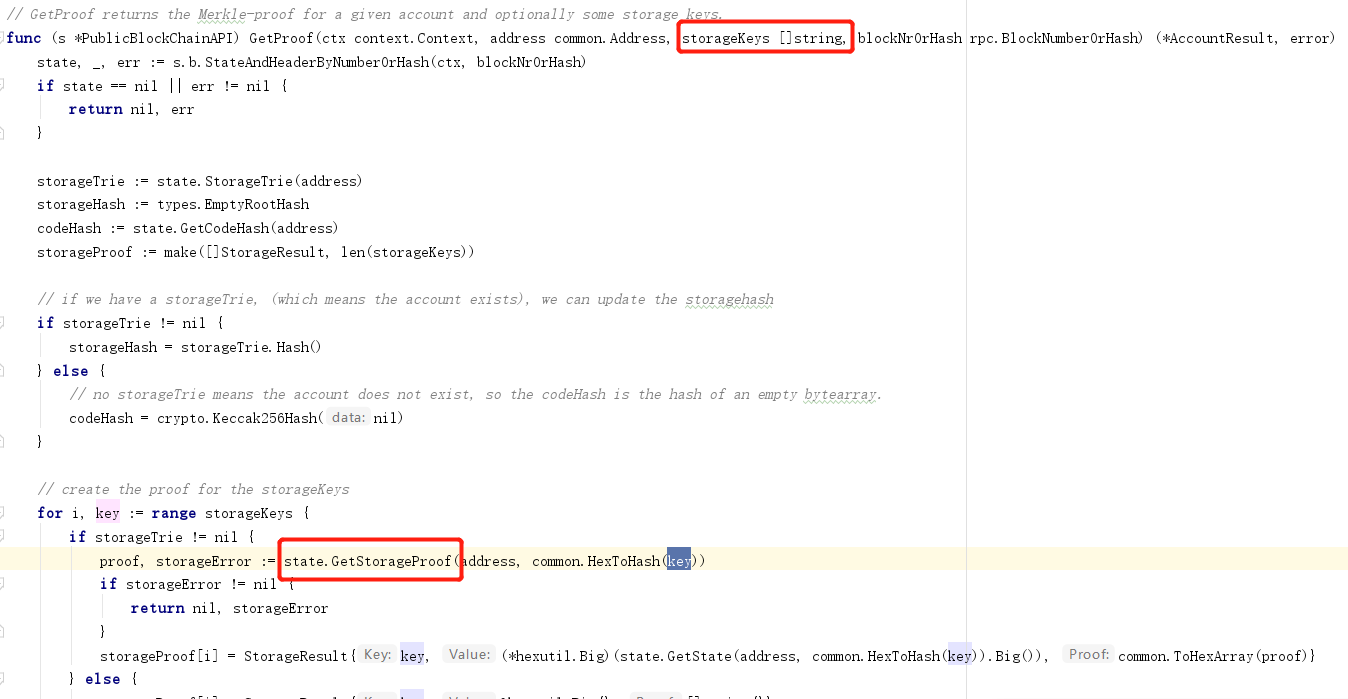
这样就去掉了最前端的0.

**深度分析：**

以太state中存储数据使用的key为hash类型，还可以以32长度的[]byte或者64长度的hex string来表示key。

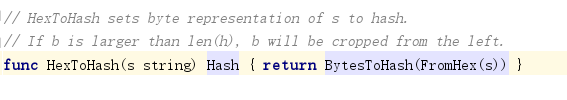
查询输入的是64长度的hex string的key，以太在查询数据之前需要转型为hash类型。

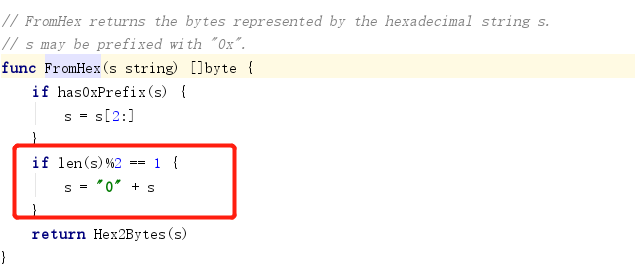
那么最前端0被去掉的并非64长度的hex string的key，以太是如何转型为hash类型的？



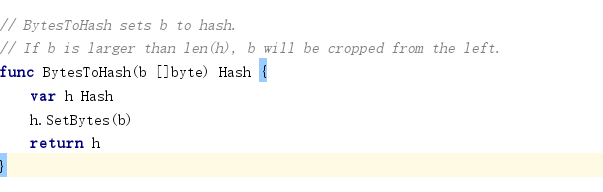
以上是以太以key来查询数据（proof）的过程，输入的key是最前端去0的hex string的key，并非一定是64的长度。

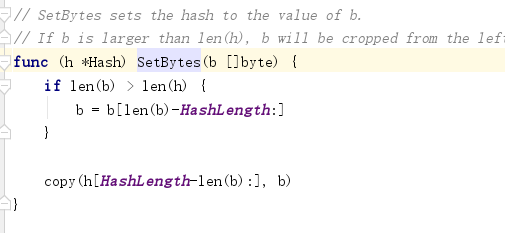
GetStorageProof是做查询操作，需要将hex string的key转型为hash类型的key来做查询。





以上是hex string转型为[]byte过程，如果长度为奇数，则补0为偶数长度，这样可以转型为[]byte。





最后将[]byte类型转型为hash类型，来查询存储。