以太坊核心存储结构分析

作者：李康

以太坊核心存储结构：Merkle-Patricia-Tree（前缀树与默克尔树的结合体），以太坊中的交易树、交易收据树、账户树以及合约存储树均使用该树索引。

该树分为三种类型节点：branch（分支，17个元素的元组）、extension（扩展，2个元素的元组）、leaf（叶子节点，2个元素的元组），因此为了区分 extension 与 leaf 节点，使用 key 的第一个 16 进制字符，其中 0000 与 0001 均代表扩展节点，0010 与 0011 均代表叶子节点，也就是说使用倒数第二位来区分 extension 与 leaf 节点。最后一位的 0、1 分别表示了该 key 原先为偶数个 16 进制字符与奇数个 16 进制字符，也就意味着为 0 时，需要填充另外的0000。

账户在以太坊中的存储

下面来看一下以太坊底层存储中是如何实现账户存储的：

```
> eth.getBlock(eth.blockNumber)
{
  difficulty: 131712,
  extraData: "0xd683010601846765746885676f312e378664617277696e",
  gasLimit: 124369302,
  gasUsed: 0,
  hash: "0xcfeda305fb42deb3467614b55d72c50663cb45a7392ab63f7f5b8b745f4e26e0",
  logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  miner: "0xa3ac96fbe4b0dce5f6f89a715ca00934d68f6c37",
  mixHash: "0x2d47b132f15e70c24ccaea79c375c3e0c601bf68ceef8247c1a911784afd6285",
  nonce: "0x0f1478e5d4e788ca",
  number: 78,
  parentHash: "0x198f2c19c6002edd5ae03aed3a265facc7ae2fa534e83c7ad46268ea4bd4fb16",
  receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 535,
  stateRoot: "0xe87cc53416653b14487af65b359e6e9a672fd1847fb199404a702f907ed4d1c3",
  timestamp: 1504495580,
  totalDifficulty: 10500310,
  transactions: [],
  transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  uncles: []
}
```

目前以太坊中存在两个账户： 0xa3ac96fbe4b0dce5f6f89a715ca00934d68f6c37 0x0f5578914288da3b9a3f43ba41a2e6b4d3dd587a

通过使用编写的拉取以太坊底层存储的 python 脚本，拉取目前底层存储的账户数据（脚本的具体用法有所改变，详见这里）：

```
➜ /Users/likang/private_ethereum  >>python detect_internal_storage.py e87cc53416653b14487af65b359e6e9a672fd1847fb199404a702f907ed4d1c3
727522965583e311a2ddb8b3499955ef767018cc1dcb47922697fc481f31bb82 :  ['\x01', '\x15\x16u\x9e\xfdZ\xf4\x00\x00', 'V\xe8\x1f\x17\x1b\xccU\xa6\xff\x83E\xe6\x92\xc0\xf8n[H\xe0\x1b\x99l\xad\xc0\x01b/\xb5\xe3c\xb4!', "\xc5\xd2F\x01\x86\xf7#<\x92~}\xb2\xdc\xc7\x03\xc0\xe5\x00\xb6S\xca\x82';{\xfa\xd8\x04]\x85\xa4p"]

1e699895f8ce1444f21dfa7bf05acc21a1cc166c1ae0169f154699f9a2933683 :  ['', '\r\xe0\xb6\xb3\xa7d\x00\x00', 'V\xe8\x1f\x17\x1b\xccU\xa6\xff\x83E\xe6\x92\xc0\xf8n[H\xe0\x1b\x99l\xad\xc0\x01b/\xb5\xe3c\xb4!', "\xc5\xd2F\x01\x86\xf7#<\x92~}\xb2\xdc\xc7\x03\xc0\xe5\x00\xb6S\xca\x82';{\xfa\xd8\x04]\x85\xa4p"]

➜ /Users/likang/private_ethereum  >>
```

其中前一项是以太坊在底层中真正存储的 key，与账户地址的对应关系如下：