实验地点： 4b201 学号: 2111605074 姓名: 马驭时 日期： 20231031

|  |  |  |  |
| --- | --- | --- | --- |
| 实验题目：  实验三：简单支付认证（SPV）实验 | | | |
| 实验目的：   1. 掌握Merkle树和SPV的理论知识以及两者的关系； 2. 掌握SPV节点通过Merkle路径快速定位交易； 3. 掌握SPV的golang代码实现； 4. 了解SPV的基本应用。 | | | |
| 实验原理及内容：   1. SPV的基本概念   我们知道比特币中区块结构分成两部分，一个是区块头，包含区块的必要属性，仅80字节大小。另一个是区块体，包含当前区块下的所有交易，一般一个区块包含成百上千笔交易，每笔交易一般要400多字节。  IMG_256  在SPV节点上，不保存全部区块链数据，不下载区块全部交易，只保存区块头数据。所以这种节点不能验证全部交易，只能用于验证支付（确认支付是在区块链中，以及确认多少次）。那么，假设一个场景：用户A在购买商品时通过比特币支付，并声称自己已经转了比特币给商家B，到商家B验证支付有效（SPV验证），过程如下：   1. SPV节点如果只关心某个支付到自己比特币地址的交易，则可以通过建立布隆过滤器（布隆过滤器是一种基于哈希的高效查找结构，能够快速确定某个元素是否在一个集合内）限制只接收含有目标比特币地址的交易。 2. 一旦比特币网络中其他当节点探测到某个交易符合SPV节点设置的布隆过滤器条件时，其它节点将以Merkleblock消息的形式发送该区块，Merkleblock消息包含区块头和一条连接目标交易与Merkle根的Merkle路径。 3. SPV节点需要验证交易，需要做2个检查，分别是：交易的存在性检查和交易是否重花的检查。 4. SPV节点通过该Merkle路径找到跟该交易相关的区块，并验证对应区块中是否存在目标交易（该过程也被称为：Merkle Path Proof）。SPV节点所收到的Merkleblock数据量通常少于1KB，只有一个完整区块（大约1MB）大小的千分之一左右。 5. 现在通过Merkle Path Proof，SPV节点确认了交易确实存在于区块链中，但是这个还是无法保证这笔交易（Transaction）的Input（引用的上一笔UTXO）没有被重花（双重支付）。这时候SPV节点通过去看这笔交易所在区块之后的区块个数，Block个数越多说明该区块被全网更多节点共识，一般来说，一笔交易所属区块之后的区块个数达到6个时，就说明这笔交易是被大家核准过（达成共识）的，没有重花，而且被篡改的可能性也很低，如下图所示：   IMG_257   1. Merkle路径  * Mekle路径是指从一笔交易到merkle root途经的所有节点。 * 验证路径是指除merkle root外，在merkle路径上所有的兄弟节点。 * 当获取到验证路径时，使用交易的hash可以计算出merkle root。     merkle证明是根据验证路径，验证一笔交易是否存在的过程。具体的验证过程与上述验证路径中说到的过程一致，如果可以根据交易的hash和验证路径计算出的merkle root与获取到的merkle root一致，则说明该笔交易确实存在于区块中。  如上图，对于交易64对应的10号节点，  其Mekle路径=（10，4，1）  验证路径=（9，3，2）  Merkle证明=（1FXq, ec20,8f74）  Merkel根=（6c0a)  SPV验证：交易的hash和验证路径计算出的merkle root与获取到的merkle root一致。  --------------------------------------------------------  实验要求：输入或读入包含n=64项的交易（64项交易用’a’-’z’,’A’-’Z’,’0’-’9’,’!’,’@’表示)，生成对应的Merkle树，对于给定的交易如‘c’，显示其交易路径，给出验证路径和Merkel证明，并进行SPV验证： | | | |
| 实验步骤及结果：  package main  import (  "fmt"  "hash/fnv"  "strings"  )  type Node struct {  Left \*Node  Right \*Node  Hash string  }  func NewNode(left, right \*Node) \*Node {  if left == nil && right == nil {  return nil  }  if left == nil {  return &Node{Right: right}  }  if right == nil {  return &Node{Left: left}  }  hash := fnv.New32a()  hash.Write([]byte(left.Hash))  hash.Write([]byte(right.Hash))  return &Node{Left: left, Right: right, Hash: fmt.Sprintf("%x", hash.Sum32())}  }  func BuildMerkleTree(transactions []string) \*Node {  if len(transactions) == 0 {  return nil  }  if len(transactions) == 1 {  return &Node{Hash: transactions[0]}  }  var nodes []\*Node  for \_, tx := range transactions {  nodes = append(nodes, &Node{Hash: tx})  }  for len(nodes) > 1 {  var newNodes []\*Node  for i := 0; i < len(nodes); i += 2 {  if i+1 < len(nodes) {  newNodes = append(newNodes, NewNode(nodes[i], nodes[i+1]))  } else {  newNodes = append(newNodes, nodes[i])  }  }  nodes = newNodes  }  return nodes[0]  }  func FindMerklePath(node \*Node, target string) ([]string, bool) {  if node == nil {  return nil, false  }  if node.Hash == target {  return []string{target}, true  }  pathLeft, foundLeft := FindMerklePath(node.Left, target)  if foundLeft {  return append([]string{node.Hash}, pathLeft...), true  }  pathRight, foundRight := FindMerklePath(node.Right, target)  if foundRight {  return append([]string{node.Hash}, pathRight...), true  }  return nil, false  }  func main() {  transactions:=strings.Split("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@", "")  merkleTree := BuildMerkleTree(transactions)  target := inputTransaction() // 获取用户输入的交易哈希值  merklePath, found := FindMerklePath(merkleTree, target)  if found {  fmt.Println("Merkle路径：", strings.Join(merklePath, " -> "))  } else {  fmt.Println("未找到交易")  }  }  func inputTransaction() string {  fmt.Print("请输入要查询的交易哈希值：")  var input string  fmt.Scanln(&input)  return input  } | | | |
| 实验小结：  通过本次实验，我对Merkle树和SPV有了更深入的了解。本次实验，我对Merkle树和SPV有了更加全面的认识，同时也锻炼了自己的编程能力和逻辑思维能力。在今后的学习和工作中，我将继续努力，不断提高自己的专业素养。 | | | |
| 成绩 |  | 评阅老师 |  |