山东大学 软件 学院

Python密码学编程课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202100150209 | 姓名： 杨佳庆 | | 班级： 21级网安班 |
| 实验题目：实验1 | | | |
| 实验学时：2 | | 实验日期： 2023.9.26 | |
| 实验目的：   * 学会使用hashlib模块实现对给定消息，利用不同的哈希算法生成消息摘要； * 使用Cryptography和Crypto库实现不同的哈希算法生成消息摘要 * 体会哈希函数的抗碰撞性； * 了解哈希函数在口令保护和区块链中的应用 | | | |
| 硬件环境：  PC机 | | | |
| 软件环境：  PyCharm | | | |
| 实验步骤与内容：   1. 编程实现生成空字符串、‘Alice’、‘Bob’的md5、sha256的哈希值；   import hashlib  def hash\_md5():  alice = hashlib.md5(b"Alice")  bob = hashlib.md5(b"Bob")  print('"Alice" md5:' + alice.hexdigest())  print('"Bob" md5:' + bob.hexdigest())  def hash\_sha256():  alice = hashlib.sha256(b"Alice")  bob = hashlib.sha256(b"Bob")  print('"Alice" sha256:' + alice.hexdigest())  print('"Bob" sha256:' + bob.hexdigest())   1. 编程实现生成自己名字的哈希值，注意编码的转换；   def myhash(str = "Yangjiaqing"):  res = hashlib.md5(str.encode(encoding="utf-8"))  print(str + " md5: " + res.hexdigest())   1. 编写体现哈希雪崩的代码，哈希值用二进制表示； 参考Listing2-5 代码；   def cmpcount(str1, str2):  count = 0  for i in range(0, len(str1)):  if str1[i] != str2[i]:  count += 1  return count  def avalanche(str1 = 'bob', str2 = 'aob'):  bin1 = str1.encode('utf-8')  bin2 = str2.encode('utf-8')  hexstring1 = hashlib.md5(bin1).hexdigest()  binstring1 = '{:08b}'.format(int(hexstring1, 16))  # binstring1 = bin(int(hexstring1, 16))  hexstring2 = hashlib.md5(bin2).hexdigest()  binstring2 = '{:08b}'.format(int(hexstring2, 16))  # binstring2 = bin(int(hexstring2, 16))  print(int(hexstring1, 16))  print(int(hexstring2, 16))  print(str1 + " md5:" + binstring1)  print(str2 + " md5:" + binstring2)  print("两个哈希值不同的位数：" + str(cmpcount(binstring1, binstring2)))   1. 利用scrypt密钥派生函数，实现口令加盐，生成更加安全的密钥（口令）；   Salt 用使用 secrets 模块生成，也可以用os.urandom()随机生成  def hash\_password(password = b'p@$Sw0rD~7'):  salt\_length = 16  salt = secrets.token\_bytes(salt\_length)  key = pyscrypt.hash(password, salt, 2048, 8, 1, 32)  return key.hex()   1. 实现区块链中的工作量证明编程，通过设置不同的难度，体会生成符合要求哈希值需要时间长短的不同；   这里的\_n\_bits就是难度值。 在比特币中，当一个块被挖出来以后，“n\_bits” 代表了区块头里存储的难度，也就是开头有多少个 0。这里的 16 指的是算出来的哈希前 16 位必须是 0  class ProofOfWork(object):  \_N\_BITS = 20  MAX\_BITS = 256  MAX\_SIZE = sys.maxsize  def \_\_init\_\_(self, n\_bits=\_N\_BITS):  self.\_n\_bits = n\_bits  self.\_target\_bits = 1 << (self.MAX\_BITS - n\_bits)  准备用于哈希的数据，为简便起见，没有用区块，只用了nonce计数器值做数据。区块链是要把区块头的数据进行合并作为计算哈希的原数据。  def \_prepare\_data(self, nonce):  data\_lst = [str(nonce)]  return encode(''.join(data\_lst))  寻找nonce的方法  def run(self):  nonce = 0  found = False  hash\_hex = None  print('Mining a new block')  while nonce < self.MAX\_SIZE:  data = self.\_prepare\_data(nonce)  hash\_hex = sum256\_hex(data)  hash\_val = int(hash\_hex, 16)  sys.stdout.write("try nonce == %d hash\_hex == %s \r" % (nonce, hash\_hex))  if (hash\_val < self.\_target\_bits):  found = True  break   nonce += 1  if found:   print('Found nonce == %d' % nonce)  else:  print('Not Found nonce')  raise NonceNotFoundError('nonce not found')  return nonce, hash\_hex  结果如下图：  补充结果截屏  1.    2.    3.程序运行时发现一个错误，是因为在16进制转2进制中出现的位数不一致造成的，因为没有直接16进制转2进制的函数，中间需要先转成10进制，造成数值大小的差位，再转2进制时前面不会自动补充0导致比较时溢出数组。对于位数一致的情况是可以正常运行的。      4.    5. | | | |
| 结论分析与体会：  在这个实验中，我们学习了哈希编程的相关知识，并使用了不同的库和算法来实现哈希函数的应用。  首先，我们使用hashlib模块实现了对给定消息的md5和sha256哈希值的生成。通过编程实现生成空字符串、'Alice'、'Bob'的哈希值和自己名字的哈希值，我们可以看到不同的输入会产生不同的哈希值。这体现了哈希函数的抗碰撞性，即不同的输入会产生不同的输出，即使输入只有微小的差异。  还编写了体现哈希雪崩效应的代码，使用二进制表示哈希值。哈希雪崩效应指的是当输入数据发生微小变化时，哈希值会发生巨大的变化。通过实验，我们可以观察到即使输入数据只有一位的变化，哈希值也会有很大的不同，这体现了哈希函数的敏感性和安全性。  此外，我们使用了scrypt密钥派生函数来实现口令加盐，生成更加安全的密钥。加盐可以增加密码的复杂度，提高密码的安全性。通过实验，我们可以体会到加盐对密码的保护作用，生成的密钥更加难以破解。  最后，我们实现了区块链中的工作量证明编程，通过设置不同的难度来体会生成符合要求哈希值所需的时间长短的不同。工作量证明是区块链中保证区块安全性的重要机制，通过实验我们可以感受到增加难度会显著增加计算的时间和资源消耗。  通过完成这个实验，我们深入了解了哈希函数的原理和应用。我们学会了使用不同的库和算法来实现哈希编程，并体会了哈希函数的抗碰撞性、哈希雪崩效应以及在口令保护和区块链中的应用。这个实验对我们的学习和职业发展都有很大的帮助，让我们更好地理解了密码学和区块链的相关概念和技术。 | | | |