project作业报告

202100460059 姜万瑞

Project 1：

直接运行即可，但时间稍长

Project 2：

# 实现方式：

因为sm3与MD5和SHA-1类似，属于迭代压缩函数类型的哈希函数，所以可以使用rho碰撞攻击，选择两个不同的随机输入，并用哈希函数进行迭代压缩。第一个输入每次迭代一次，第二个每次迭代两次，然后比较每次的hash输出，找到碰撞，这种方法减少了存储空间。

# 实验环境

Windows11

visual studio 2022

CPU：11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz

# 实现效果

运行速度0.824s

Project 3：

在知晓待hash的信息长度前提之下，在信息之后填充一定数量的零，直达信息长度为分组长度的整数倍，然后再添加自定义的信息，根据MD结构，这样操作之后的信息的hash值可以由以原先信息的hash值作为IV，新自定义信息作为待hash信息得到。我们将这样操作之后得到的hash值与plaintext||000...000||data\_append的hash值做对比，若相同，则说明正确。

# 实验环境

Windows11

visual studio 2022

CPU：11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz

# 实现效果

运行速度0.003s

Project 4：

看文件

Project 5：

本次实验中，我们先用random.randint生成10万的数据作为区块数据，再据此生成merkle tree。

当我们检查或者证明某个节点是否存在时，只需给出该节点的相邻兄弟节点和父节点即可，具体方式是对于待证明的节点，我们只需要确切找出它的hash值是否与树中某节点hash值相同以及该节点在树中的位置，可以求解其在list中对应的下标n；

我们需要讨论待求是偶节点还是奇节点。根据其下标，如果是奇数：首先检查是否是最后一个节点。 如果是，则不操作，进入下一个循环。

找到该节点之后，将其右节点（下标+1的节点）与该节点一起写入列表。 如果是偶数：将其左节点（下标为-1的节点）与该节点一起写入列表；之后我们就可反复地调用上面的程序，最后将根节点放入树中。

验证是否其父节点的hash值等于hash（子节点的hash值之和），然后再照此验证其祖父节点。此外我们还应该检查该树是否符合merkle tree的定义结构。

# 实验环境

Windows11

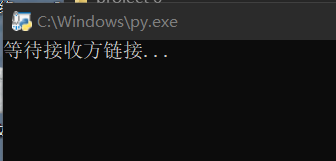
PYCHARM 2022

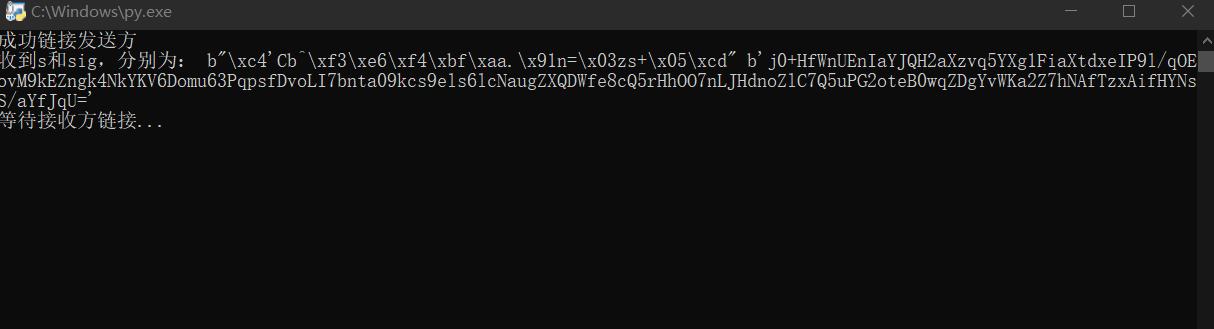
CPU：11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz

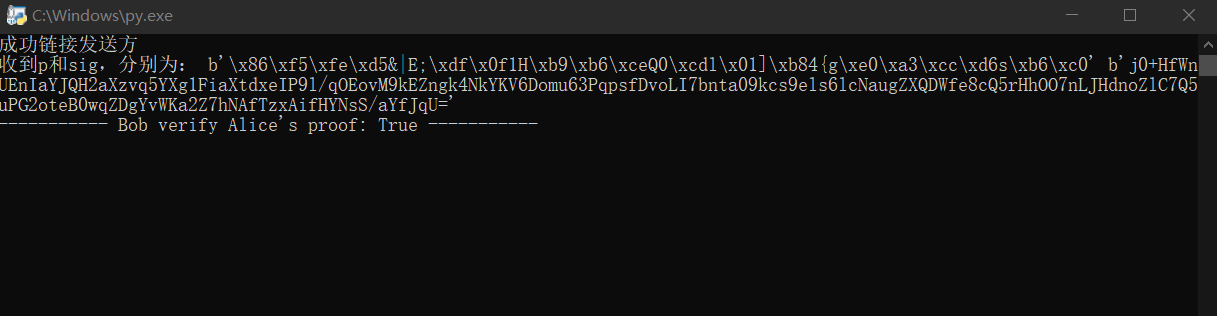
Project 6：

实现方式：该例子主要是应用到了hash-chain和签名。hash-chain即对某一对象多次hash，在本实验实现中，因hash函数接受的是字节流，所以对hash后的结果进行.digest()即可得到字节流。本实验的两个hash函数分别为SHA1和SHA256，签名为RSA签名。

实现效果：







Project 7：

主要还是使用了与上一个project相同的hash-chain。在开始时，还需要使用一个KDF，我们根据线性同余法编写自己的KDF。还有一个shuffle()函数，根据一个shuffle种子，然后初始化随机数生成器，再使用random.shuffle()对哈希值列表进行shuffle即可。

Project 9：

详细看文件

Project 10：

详细看文件

Project 11：

为了避免私钥泄露，本次实验中，我们令k=hash（随机数||ID||"算法类型"），符合RFC 6979的确定性随机数生成算法，以在不依赖真正随机数的情况下根据消息和私钥生成一系列伪随机数。

此外Python语言实现了SM2算法

实验结果

运行速度：0.05s

实验环境

Windows11

pycharm 2022

CPU：11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz

Project 12：

详细看文件

Project 17：

区别：

在声明权限时，Firefox使用"permissions"字段，而谷歌浏览器使用"permissions"字段或"optional\_permissions"字段来声明不同类型的权限。

firefox:

"manifest\_version": 2,

"name": "Your Password Manager",

"version": "1.0",

"description": "A password manager extension for Firefox",

"permissions": [

"tabs",

"storage"

],

"content\_scripts": [

{

"matches": ["<all\_urls>"],

"js": ["content.js"]

}

],

"browser\_action": {

"default\_icon": "icon.png",

"default\_popup": "popup.html"

}

google:

"manifest\_version": 2,

"name": "Your Password Manager",

"version": "1.0",

"description": "A password manager extension for Chrome",

"permissions": [

"tabs",

"storage"

],

"content\_scripts": [

{

"matches": ["<all\_urls>"],

"js": ["content.js"]

}

],

"browser\_action": {

"default\_icon": "icon.png",

"default\_popup": "popup.html"

}

在获取当前选项卡信息时，Firefox使用'browser.tabs' API，而谷歌浏览器使用'chrome.tabs' API。

firefox:

browser.tabs.query({ currentWindow: true, active: true }, function(tabs) {

const currentTab = tabs[0];

});

google:

chrome.tabs.query({ currentWindow: true, active: true }, function(tabs) {

const currentTab = tabs[0];

});

对登录表单提交事件的监听和内容脚本的注入的不同

firefox:

document.addEventListener("submit", function(event) {

});

google:

document.addEventListener("submit", function(event) {

});

Project 21：

方式：按照Schnorr签名算法，因需要很多计算，所以在caculate.py文件中写了离散对数的求解，素数和素因子的生成以及计算生成元。这样就可以在Schnorr.py文件中直接使用这些计算。