***Project1***: implement the naïve birthday attack of reduced SM3

**思路：**

1. 首先导入所需的库和模块。

2. 定义函数 sm3\_hash(message)。

3. 定义函数 generate\_collision()（生成两个具有相同哈希值的消息。）

4. generate\_collision() 函数，首先随机生成一个message，然后调用 sm3\_hash(message)得到其哈希值 hash\_value。

5. 初始化一个messages，存储所生成的消息。

6. 找到两个具有相同哈希值的消息停止循环。

7. 在循环中，随机生成另一个消息 new\_message并调用 sm3\_hash(new\_message)得到其哈希值 new\_hash\_value。

8. 将新的消息 new\_message添加到 messages 列表中。

9. 检查新的哈希值 new\_hash\_value 如果在 messages 列表中出现过就停止循环并返回 message 和 new\_message。

10. 如果没有则将新的哈希值 new\_hash\_value添加到 messages列表中。

**运行时间：**取决计算机性能和运气，时间从几秒钟到几分钟不等。

***Project2***: implement the Rho method of reduced SM3

**思路：**

1. 首先将输入的消息转换为字节数组，并初始化一个32字节的数组h。
2. 对输入消息进行迭代处理，每64个字节为一轮。在每一轮中，程序将该轮的字节转换为64位和128位的二进制数，并使用Chang背着程求解。
3. 使用Rho方法进行轮函数变换，并将结果与h进行异或运算。
4. 将结果左移5位，并将h转换为十六进制字符串，作为该轮的哈希值输出。

**运行时间：**运行时间取决于输入消息的长度和计算机的性能。对于长度为1MB的消息，运行时间大约在几秒钟左右。

***Project3***: implement length extension attack for SM3, SHA256, etc.

**思路：**

1.构造一个消息M，使其哈希值的前缀与目标消息的前缀相同。

2.将目标消息拆分为两个部分：已知前缀和目标后缀。

3.使用SM3或SHA256对构造的消息进行哈希计算，得到其哈希值。

4.对哈希值进行位移操作，使其成为已知前缀和目标后缀的拼接结果。

5.对位移后的哈希值进行消息扩展操作，得到扩展后的消息。

6.使用SM3或SHA256对扩展后的消息进行哈希计算，得到其哈希值。

7.比较扩展后的消息的哈希值和目标消息的哈希值，如果相同，则攻击成功。

**运行时间：**同样取决于消息长度，以1MB消息为例，时间在0.5s左右。

***Project4***: do your best to optimize SM3 implementation (software)

**思路：**

1. 将输入的消息分块，每块64字节，并将每个块转换为8个64位整数
2. 对W数组进行一系列的异或和左移操作，并更新W数组的值
3. 对哈希值进行80轮迭代计算，每轮包括以下操作：

将C左移63位，并将结果与C异或。

计算T = (E左移56位) ^ ((A & B) ^ (C & (A | D))) ^ D ^ W[i] ^hash\_buf[i]。

更新hash\_buf[i]、E、D、C、B、A的值。

1. 将计算得到的哈希值与初始哈希值进行异或，并输出最终的哈希值。

**运行时间：**

对于64字节的输入消息，时间为1ms左右。

***Project5***: Impl Merkle Tree following RFC6962

**思路：（参考网络go语言）**

1.hash 函数接受一个字节数组作为输入，使用 SHA-256 哈希算法计算其哈希值，并返回计算结果。

2.proof 函数接受一个数据字节数组和一个目标哈希值作为输入，计算出 Merkle Tree 的根哈希值，并在树中查找是否存在一个叶子节点的哈希值等于目标哈希值。如果存在，则返回证明存在该叶子的证据，否则返回证据不存在。该函数通过在每一层进行尝试来查找正确的路径，然后构建出证明。如果找到了正确的叶子节点，证明存在；否则，证明不存在。

3.bytesEqual 函数用于比较两个字节数组是否相等。

**运行时间：**

对于一个包含 n 个节点的 Merkle Tree，最坏情况下，需要遍历 n 个叶子节点和 n-1 个内部节点才能找到目标哈希值所对应的节点。因此，最坏情况下的运行时间复杂度为 O(n)。

***Project14***: Implement a PGP scheme with SM2

**思路：**

1.导入相关的库：需要导入公钥和私钥的生成库，以及加密和解密库。

2.生成密钥对：使用SM2算法生成公钥和私钥。

3.加密明文：使用公钥对明文进行加密。

4.解密密文：使用私钥对密文进行解密。

**运行时间：**无法准确预估

***Project17***：比较Firefox和谷歌的记住密码插件的实现区别

Firefox和谷歌浏览器都提供了记住密码的功能，但是它们的实现方式略有不同。

Firefox的记住密码功能是由Password Manager 负责管理，它会自动保存用户名和密码，并且可以在需要时自动填充。

谷歌浏览器的记住密码功能是通过自动填充表单实现的。