*Project2: implement the Rho method of reduced SM3

代码说明：

此项目是实现 SM3 的 Rho 攻击。

Rho 攻击是一种基于碰撞概率的攻击方法，适用于部分哈希函数。这种攻击方法利用了哈希函数的迭代结构和碰撞概率，通过找到两个具有相同哈希值的输入来构造攻击。

基本原理是通过选择不同的初始输入值，并跟踪哈希函数迭代过程中产生的中间值，以期望找到具有相同中间值的两个输入，进而产生哈希碰撞。

实现 SM3 的前 n 位 Rho 攻击：

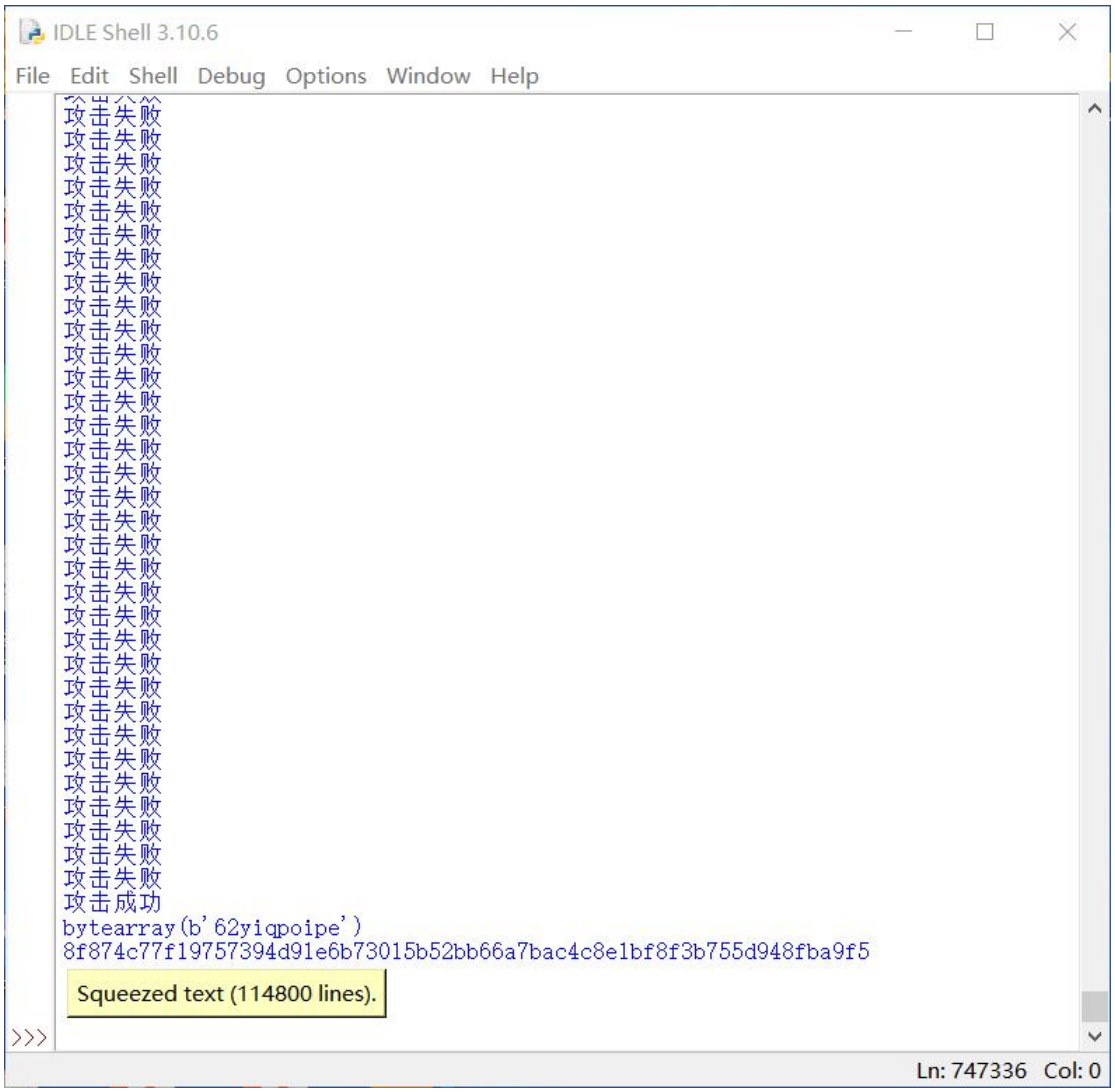
定义 generate_random_string 函数，接受一个参数 length，生成一个长度为 length 的随机字符串。定义 rho_attack 函数，接受一个参数 n，将 n 转换为前 n 位比特数。然后，通过 for 循环迭代生成大量随机字符串 b，并计算其 SM3 哈希值的前 n 位比特 c。检查 c 是否在列表 a 中，如果在，则认为攻击成功，并打印出成功信息以及找到的两个不同字符串和哈希值。如果不在，将 c 添加到列表 a 中，并继续生成新的随机字符串 b。

注：rho_attack(n) 中可根据需要将函数变量 n 改为任意整数。

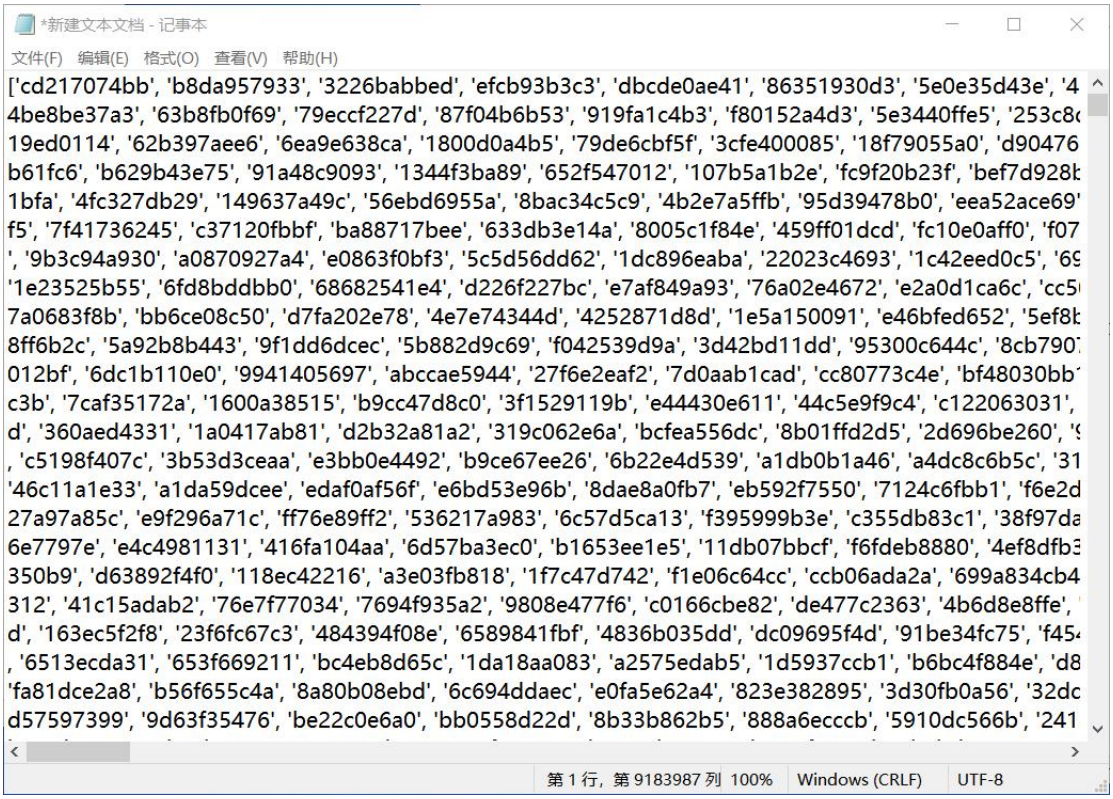
实现方式：python

效果：在自己电脑上 CPU：11 代 i7

这里以 rho_attack(40)为例，展示其结果：



Squeezed text(114800 lines)中的内容：


*新建文本文档 - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
['cd217074bb', 'b8da957933', '3226babbed', 'efcb93b3c3', 'dbcde0ae41', '86351930d3', '5e0e35d43e', '4
4be8be37a3', '63b8fb0f69', '79eccf227d', '87f04b6b53', '919fa1c4b3', 'f80152a4d3', '5e3440ffe5', '253c8
19ed0114', '62b397aee6', '6ea9e638ca', '1800d0a4b5', '79de6cbf5f', '3cfe400085', '18f79055a0', 'd90476
b61fc6', 'b629b43e75', '91a48c9093', '1344f3ba89', '652f547012', '107b5a1b2e', 'fc9f20b23f', 'bef7d928
1bfa', '4fc327db29', '149637a49c', '56ebd6955a', '8bac34c5c9', '4b2e7a5ffb', '95d39478b0', 'eea52ace69'
f5', '7f41736245', 'c37120fbbf', 'ba88717bee', '633db3e14a', '8005c1f84e', '459ff01dcd', 'fc10e0aff0', 'f07
', '9b3c94a930', 'a0870927a4', 'e0863f0bf3', '5c5d56dd62', '1dc896eaba', '22023c4693', '1c42eed0c5', '69
'1e23525b55', '6fd8bddbb0', '68682541e4', 'd226f227bc', 'e7af849a93', '76a02e4672', 'e2a0d1ca6c', 'cc5
7a0683f8b', 'bb6ce08c50', 'd7fa202e78', '4e7e74344d', '4252871d8d', '1e5a150091', 'e46bfed652', '5ef8
8ff6b2c', '5a92b8b443', '9f1dd6dcec', '5b882d9c69', 'f042539d9a', '3d42bd11dd', '95300c644c', '8cb790
012bf', '6dc1b110e0', '9941405697', 'abccae5944', '27f6e2eaf2', '7d0aab1cad', 'cc80773c4e', 'bf48030bb
c3b', '7caf35172a', '1600a38515', 'b9cc47d8c0', '3f1529119b', 'e44430e611', '44c5e9f9c4', 'c122063031',
d', '360aed4331', '1a0417ab81', 'd2b32a81a2', '319c062e6a', 'bcfea556dc', '8b01ffd2d5', '2d696be260', '
, 'c5198f407c', '3b53d3ceaa', 'e3bb0e4492', 'b9ce67ee26', '6b22e4d539', 'a1db0b1a46', 'a4dc8c6b5c', '31
'46c11a1e33', 'a1da59dcee', 'edaf0af56f', 'e6bd53e96b', '8dae8a0fb7', 'eb592f7550', '7124c6fbb1', 'f6e2d
27a97a85c', 'e9f296a71c', 'ff76e89ff2', '536217a983', '6c57d5ca13', 'f395999b3e', 'c355db83c1', '38f97da
6e7797e', 'e4c4981131', '416fa104aa', '6d57ba3ec0', 'b1653ee1e5', '11db07bbcf', 'f6fdeb8880', '4ef8dfb3
350b9', 'd63892f4f0', '118ec42216', 'a3e03fb818', '1f7c47d742', 'f1e06c64cc', 'ccb06ada2a', '699a834cb4
312', '41c15adab2', '76e7f77034', '7694f935a2', '9808e477f6', 'c0166cbe82', 'de477c2363', '4b6d8e8ffe',
d', '163ec5f2f8', '23f6fc67c3', '484394f08e', '6589841fbf', '4836b035dd', 'dc09695f4d', '91be34fc75', 'f45
, '6513ecda31', '653f669211', 'bc4eb8d65c', '1da18aa083', 'a2575edab5', '1d5937ccb1', 'b6bc4f884e', 'd8
'fa81dce2a8', 'b56f655c4a', '8a80b08ebd', '6c694ddaec', 'e0fa5e62a4', '823e382895', '3d30fb0a56', '32dc
d57597399', '9d63f35476', 'be22c0e6a0', 'bb0558d22d', '8b33b862b5', '888a6ecccb', '5910dc566b', '241
第 1 行，第 9183987 列 100%
Windows (CRLF)
UTF-8

分工：自己独立完成