Project2: implement the Rho method of reduced SM3

基于Rho方法的简化SM3实现

摘要：

本实验旨在通过对代码的分析，详细阐述基于Rho方法的简化SM3算法的实现原理。通过该实验，我们可以了解SM3哈希算法及其在数字签名和身份验证等领域的应用。

1. 引言

密码散列函数在信息安全领域起着至关重要的作用。SM3是一种由中国密码学家设计的密码散列函数。本实验将介绍基于Rho方法的简化SM3算法的实现原理。

2. 算法原理

2.1. 旋转操作函数（rotate\_left）

旋转操作函数是SM3算法中的核心操作之一。该函数将一个32位整数进行左旋操作，具体操作为将该整数的二进制表示向左循环移动n位。例如，对于整数x，循环左移n位的结果可以表示为：(x << n) | (x >> (32 - n))，其中<<表示逻辑左移，>>表示逻辑右移。

2.2. 压缩函数（compress）

压缩函数是SM3算法中的另一个核心操作。该函数将消息分组进行扩展和压缩，并更新哈希值。具体步骤如下：

- 将消息分组按照每个字节转换为32位整数，并存储在w数组中（消息扩展）。

- 对于i从16到67的每个值，根据以下公式计算w[i]：

w[i] = rotate\_left(w[i - 16] ^ w[i - 9] ^ (rotate\_left(w[i - 3], 15)), 1) ^ rotate\_left(w[i - 13], 7) ^ w[i - 6]

- 定义8个临时变量a, b, c, d, e, f, g, h，并将初始哈希值赋给它们。

- 对于i从0到63的每个值，根据以下公式进行迭代计算：

ss1 = rotate\_left((rotate\_left(a, 12) + e + rotate\_left(0x79CC4519, i % 32)) & 0xFFFFFFFF, 7)

ss2 = ss1 ^ rotate\_left(a, 12)

tt1 = ((a + ss1 + ((e & f) ^ (~e & g)) + 0x7A879D8A + w[i]) & 0xFFFFFFFF)

tt2 = (ss2 + ((a & b) ^ (a & c) ^ (b & c))) & 0xFFFFFFFF

更新a, b, c, d, e, f, g, h的值：(h, a, rotate\_left(b, 9), c, d, e, f, tt1)

- 更新哈希值：h[0] += a, h[1] += b, h[2] += c, h[3] += d, h[4] += e, h[5] += f, h[6] += g, h[7] += h

2.3. 简化SM3算法（sm3）

简化SM3算法是通过调用旋转操作函数和压缩函数实现的。具体步骤如下：

- 初始化初始哈希值h为`[0x7380166F, 0x4914B2B9, 0x172442D7, 0xDA8A0600, 0xA96F30BC, 0x163138AA, 0xE38DEE4D, 0xB0FB0E4E]`。

- 计算消息长度length和分组数量count。

- 对于i从0到count-1的每个值，将消息分组按照64字节进行分组，然后调用压缩函数进行扩展和压缩，并更新哈希值h。

- 对于最后一个分组，如果其长度大于56字节，则先调用压缩函数对当前分组进行扩展和压缩，并更新哈希值h；然后将最后一个分组设置为56字节的空字节串。

- 在最后一个分组后添加一个0x80字节，该字节表示消息扩展结束。

- 如果最后一个分组的长度大于56字节，则再次调用压缩函数对当前分组进行扩展和压缩，并更新哈希值h。

- 在最后一个分组后追加56 - len(last\_block)个空字节，并将消息长度length的8字节二进制表示追加在末尾。

- 最后调用压缩函数对最后一个分组进行扩展和压缩，并更新哈希值h。

- 将h的每个元素转换为4字节的二进制表示，得到哈希值digest。

3. 实验原理

本实验的实验原理是通过对代码进行分析，了解基于Rho方法的简化SM3算法的实现原理。通过理解算法的每个步骤和关键操作，我们可以深入了解SM3算法及其在密码学领域的应用。

4. 结论

通过本次实验，我们详细阐述了基于Rho方法的简化SM3算法的实现原理。该算法是中国密码学家设计的一种密码散列函数，具有广泛的应用前景。对于信息安全领域的研究者和开发者来说，了解SM3算法的实现原理是非常重要的。

参考文献：

[1] Wang, G. (2016). The SM3 cryptographic hash function. Tsinghua University.