


《现代密码学》实验报告

实验名称: SHA-256	实验时间: 2022.11.05
学生姓名: 伍建霖	学号: 20337251
学生班级: 20网安	成绩评定:

一、实验目的


了解SHA256算法的基本原理，掌握SHA256算法的实现方法，完成字符串的SHA256运算及算法流程。

二、实验内容



实验要求

- 1.给出
“fr356cqEm7SOqBtvOOOx%MkR&ETUJvuE6AcYNaLSKSLlt6Y4my8I2pLDk#FEkBMopG5XtoTB6pl4kmU6DvsWDT2In5K#wPHW” + “学号” 的SHA256的结果
- 2.自己给出一个不小于2kb的文本文件，在文件末尾拼接上自身学号后进行hash,输出256比特的SHA256哈希值。作业中要附上进行hash的文本文件并命名为“test.msg”



三、实验原理

SHA256算法原理：填充——>分块——>扩展——>更新哈希值——>扩展下一块——>更新哈希值.....

四、实验步骤

常量

当前的哈希值（散列值，消息摘要）都会参与下一轮的更新，第一轮也是如此，故需要一组初始值：

```

void SHA256::init()
{
    m_h[0] = 0x6a09e667;
    m_h[1] = 0xbb67ae85;
    m_h[2] = 0x3c6ef372;
    m_h[3] = 0xa54ff53a;
    m_h[4] = 0x510e527f;
    m_h[5] = 0x9b05688c;
    m_h[6] = 0x1f83d9ab;
    m_h[7] = 0x5be0cd19;
    m_len = 0;
    m_tot_len = 0;
}

```

除了初始哈希值，我们还需要64个常量，参与更新哈希值，起到类似密钥的作用：

```

const unsigned int SHA256::sha256_k[64] =
{0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5,
 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
 0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3,
 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
 0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc,
 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
 0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7,
 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
 0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13,
 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
 0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3,
 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
 0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5,
 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
 0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208,
 0x90bffffffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2};

```

填充，分块

准备好哈希值和密钥之后，我们还需要对原始消息进行预处理：

先在消息末尾补一个"1"，使得消息长度 = $512 * k + 448$ 。

再在末尾补上64位的二进制的消息长度，使得消息长度 = $512 * k$ 。

```

// 填充
block_nb = (1 + ((SHA_256_BLOCK_SIZE - 9) < (m_len % SHA_256_BLOCK_SIZE)));
len_b = (m_tot_len + m_len) << 3;
pm_len = block_nb << 6;
memset(m_block + m_len, 0, pm_len - m_len);
m_block[m_len] = 0x80;
SHA2_UNPACK32(len_b, m_block + pm_len - 4);

// 分块相关
sub_block = message + (i << 6);
for (j = 0; j < 16; j++)
{

```

```

        SHA2_PACK32(&sub_block[j << 2], &w[j]);
    }

```

扩展

32位一个字，512位就一共有16个字，我们需要从这16个字中扩展出64个字。

```

// 扩展出64个字
#define SHA2_SHFR(x, n) (x >> n)
#define SHA2_ROTTR(x, n) ((x >> n) | (x << ((sizeof(x) << 3) - n)))
#define SHA2_ROTTL(x, n) ((x << n) | (x >> ((sizeof(x) << 3) - n)))
#define SHA256_F3(x) (SHA2_ROTTR(x, 7) ^ SHA2_ROTTR(x, 18) ^ SHA2_SHFR(x, 3))
#define SHA256_F4(x) (SHA2_ROTTR(x, 17) ^ SHA2_ROTTR(x, 19) ^ SHA2_SHFR(x, 10))

for (j = 16; j < 64; j++)
{
    w[j] = SHA256_F4(w[j - 2]) + w[j - 7] + SHA256_F3(w[j - 15]) + w[j - 16];
}

```

更新

我们最终需要的是256位的哈希值，我们已有初始的哈希值，每一块（512位）都会通过一系列运算得出一个中间值（下面的abcdefgh），这个中间值和上一块的哈希值运算得出当前块的哈希值，如 $H_1^{(1)} = a + H_1^{(0)}$ 。

$$\begin{aligned}
 H_1^{(i)} &\leftarrow a + H_1^{(i-1)} \\
 H_2^{(i)} &\leftarrow b + H_2^{(i-1)} \\
 &\vdots \\
 H_8^{(i)} &\leftarrow h + H_8^{(i-1)}
 \end{aligned}$$

```

for (j = 0; j < 8; j++)
{
    m_h[j] += wv[j];
}

```

至于如何得到中间值，就靠下面伪代码表示的更新算法：

- For $j = 0 \rightarrow 63$
 - 计算 $Ch(e, f, g)$, (具体定义如下)
 - $M_{aj}(a, b, c)$,
 - $\Sigma_0(a), \Sigma_1(e)$,
 - W_j

$$\begin{aligned}
 T_1 &\leftarrow h + \Sigma_1(e) + Ch(e, f, g) + K_j + W_j \\
 T_2 &\leftarrow \Sigma_0(a) + M_{aj}(a, b, c) \\
 h &\leftarrow g \\
 g &\leftarrow f \\
 f &\leftarrow e \\
 e &\leftarrow d + T_1 \\
 d &\leftarrow c \\
 c &\leftarrow b \\
 b &\leftarrow a \\
 a &\leftarrow T_1 + T_2
 \end{aligned}$$

```
// 更新算法
for (j = 0; j < 64; j++)
{
    t1 = wv[7] + SHA256_F2(wv[4]) + SHA2_CH(wv[4], wv[5], wv[6]) + sha256_k[j] + w[j];
    t2 = SHA256_F1(wv[0]) + SHA2_MAJ(wv[0], wv[1], wv[2]);
    wv[7] = wv[6];
    wv[6] = wv[5];
    wv[5] = wv[4];
    wv[4] = wv[3] + t1;
    wv[3] = wv[2];
    wv[2] = wv[1];
    wv[1] = wv[0];
    wv[0] = t1 + t2;
}
```

更新算法中的逻辑函数的具体含义：

$$\begin{aligned}
 Ch(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z) \\
 M_{aj}(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\
 \Sigma_0(x) &= S^2(x) \oplus S^{13}(x) \oplus S^{22}(x) \\
 \Sigma_1(x) &= S^6(x) \oplus S^{11}(x) \oplus S^{25}(x) \\
 \sigma_0(x) &= S^7(x) \oplus S^{18}(x) \oplus R^3(x) \\
 \sigma_1(x) &= S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x)
 \end{aligned}$$

```
// 逻辑函数
#define SHA2_CH(x, y, z) ((x & y) ^ (~x & z))
#define SHA2_MAJ(x, y, z) ((x & y) ^ (x & z) ^ (y & z))
#define SHA256_F1(x) (SHA2_ROT(x, 2) ^ SHA2_ROT(x, 13) ^ SHA2_ROT(x, 22))
#define SHA256_F2(x) (SHA2_ROT(x, 6) ^ SHA2_ROT(x, 11) ^ SHA2_ROT(x, 25))
```

五、实验结果

1. 给出

“fr356cqEm7SOqBtvOOOx%MkR&ETUJvuE6AcYNaLSKSLlt6Y4my8I2pLDk#FEkBMopG5XtoTB6p14kmU6DvsWDT2In5K#wPHW”+ “学号”的SHA256的结果

程序结果

```
问题  输出  调试控制台  终端

sha256('fr356cqEm7SOqBtvOOOx%MkR&ETUJvuE6AcYNaLSKSLlt6Y4my8I2pLDk#FEkBMopG5XtoTB6p14kmU6DvsWDT2In5K#wPHW20337251'):3fdd0a96d0e3b4673106919fa75af846d3744dae0871b793cf16325b68adcd68
(base) PS D:\CodeField\Cryptography\lab> & 'c:\Users\henry\.vscode\extensions\ms-vscode.cpptools-1.12.4-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-eq43vscp.qxd' '--stdout=Microsoft-MIEngine-Out-wssqj2gy.uhd' '--stderr=Microsoft-MIEngine-Error-imubgt5.upd' '--pid=Microsoft-MIEngine-Pid-dg11sgzk.1v1' '--dbgExe=C:\Program Files\mingw64\bin\gdb.exe' '--interpreter=mi'
sha256('fr356cqEm7SOqBtvOOOx%MkR&ETUJvuE6AcYNaLSKSLlt6Y4my8I2pLDk#FEkBMopG5XtoTB6p14kmU6DvsWDT2In5K#wPHW20337251'):3fdd0a96d0e3b4673106919fa75af846d3744dae0871b793cf16325b68adcd68
(base) PS D:\CodeField\Cryptography\lab> █
```

网站结果

Your generated hash

hex: 3fdd0a96d0e3b4673106919fa75af846d3744dae0871b793cf16325b68adcd68

HEX: 3FDD0A96D0E3B4673106919FA75AF846D3744DAE0871B793CF16325B68ADCD68

h:e:x: 3f:dd:0a:96:d0:e3:b4:67:31:06:91:9f:a7:5a:f8:46:d3:74:4d:ae:08:71:b7:93:cf:16:32:5b:68:ad:cd:68

base64: P90KltDjtGcxBpGfp1r4RtN0Ta4IcbeTzxYyW2itzWg=

2. 自己给出一个不小于2kb的文本文件，在文件末尾拼接上自身学号后进行hash,输出256比特的SHA256哈希值。作业中要附上进行hash的文本文件并命名为“test.msg”

程序结果

```
问题  输出  调试控制台  终端

7251fr356cqEm7SOqBtvOOOx%MkR&ETUJvuE6AcYNaLSKSLlt6Y4my8I2pLDk#FEkBMopG5XtoTB6p14kmU6DvsWDT2In5K#wPHW20337251fr356cqEm7SOqBtvOOOx%MkR&ETUJvuE6AcYNaLSKSLlt6Y4my8I2pLDk#FEkBMopG5XtoTB6p14kmU6DvsWDT2In5K#wPHW20337251fr356cqEm7SOqBtvOOOx%MkR&ETUJvuE6AcYNaLSKSLlt6Y4my8I2pLDk#FEkBMopG5XtoTB6p14kmU6DvsWDT2In5K#wPHW20337251fr356cqEm7SOqBtvOOOx%MkR&ETUJvuE6AcYNaLSKSLlt6Y4my8I2pLDk#FEkBMopG5XtoTB6p14kmU6DvsWDT2In5K#wPHW20337251'):072446e061b2995e9e8ac2149aa1e87ec31aecb721540dde3527a37d900111f9
(base) PS D:\CodeField\Cryptography\lab> █
```

网站结果

Your generated hash

hex: 072446e061b2995e9e8ac2149aa1e87ec31aecb721540dde3527a37d900111f9

HEX: 072446E061B2995E9E8AC2149AA1E87EC31AECB721540DDE3527A37D900111F9

h:e:x: 07:24:46:e0:61:b2:99:5e:9e:8a:c2:14:9a:a1:e8:7e:c3:1a:ec:b7:21:54:0d:de:35:27:a3:7d:90:01:11:f9

base64: ByRG4GGymV6eislUmqHofsMa7LchVA3eNSejfZABEfK=

六、实验总结

参考资料：

[Download file – your conversion was successful \(online-convert.com\)](#)

[一文读懂SHA256算法原理及其实现 - 知乎 \(zhihu.com\)](#)

[Sha256 Algorithm Explained](#)

从这次实验中我学习到了如何在c++中实现sha256算法，了解了其中的具体过程。