Hash函数

MD5

- 16字节,128位Hash

1.md5计算过程中的分块及填充

分块计算,每块大小为64字节(64字节data数组)

> 填充物的最后8个字节必须是报文(message)的长度，所以填充长度不同的临界点是64-8=56个字节

设最后一块message长度为n

$$\begin{cases} 填充56-n字节的0x80\ 0x00\ ...\ 0x00 & 0 \leq n < 56 \\ 填充120-n字节的0x80\ 0x00\ ...\ 0x00 & 56 < n < 64 \\ \quad (算作倒数第二块,另起一块n=0字节 & n = 64) \end{cases}$$

故而，n的真实范围是

$$n \in [0, 63)$$

最后8个字节填充报文长度

- 填充物
  二进制格式 100...0
  16进制格式 0x80 0x00 0x00 ... 0x00

```
static unsigned char pad_stuff[64] = {
0x80,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
};
```

- 报文长度
  报文长度是指报文的位长度，小端格式，共8字节

2.md5计算过程中的分块及填充

- Init_MD5

```
int Init_MD5(MD5_CTX *MD5_ctx)
```

- Update_MD5

```
int Update_MD5(MD5_CTX *MD5_ctx,
            unsigned char *buf,
            unsigned long buf_len)
```

- Process_One_Block_MD5

```
static void Process_One_Block_MD5(unsigned long state[4], unsigned char block[64])
```

  - state表示上一次process block时计算出MD5的值，每个 state 16个字节-128位Hash
  - block即为64字节的一块message
- Final_MD5

```
int Final_MD5(MD5_CTX *MD5_ctx)
```

> linux编译oppenssl
> gcc md5.c -o md5 -lcrypto

3.md5破解方法：rainbow table

SHA

- 160位，20字节Hash
- 分块计算，每块24字节