SHA256散列方式

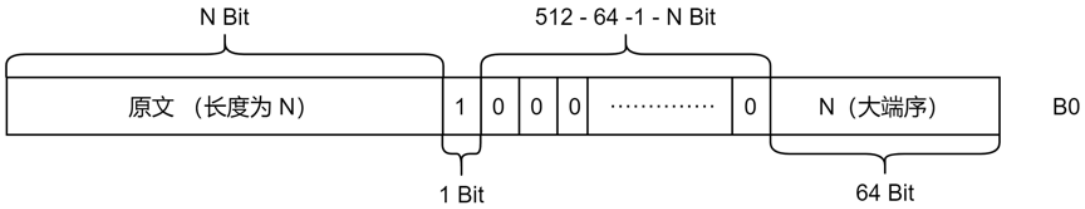
第一步

将需要散列的 16进制 字符串补充到 长度mod512 = 0 ， 参考代码：SHA256_Data 结构 的 构造函数。（同SM3）

补充方式

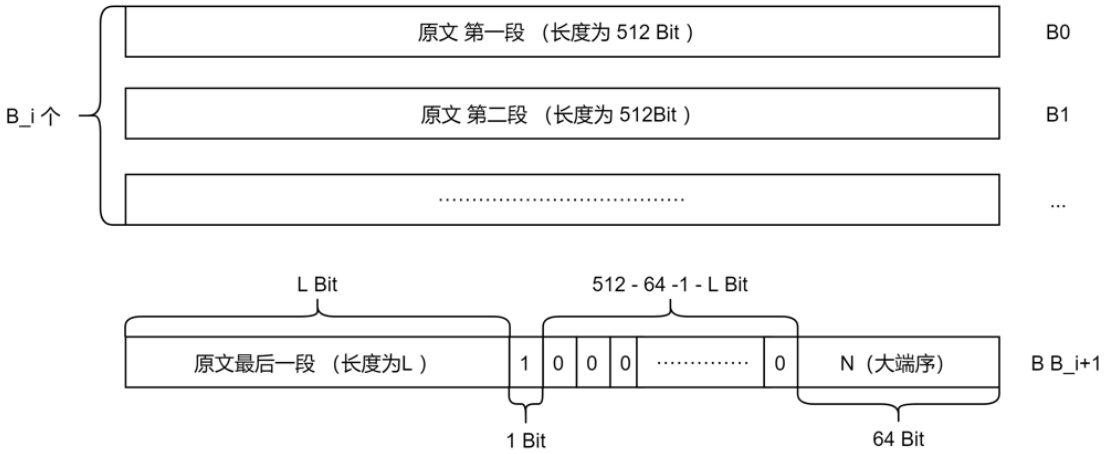
设 需要散列的 文本为"M"， M的长度为N Bit , B_i = N/512 , L = N%512。

N <= 512-64-1 Bit：

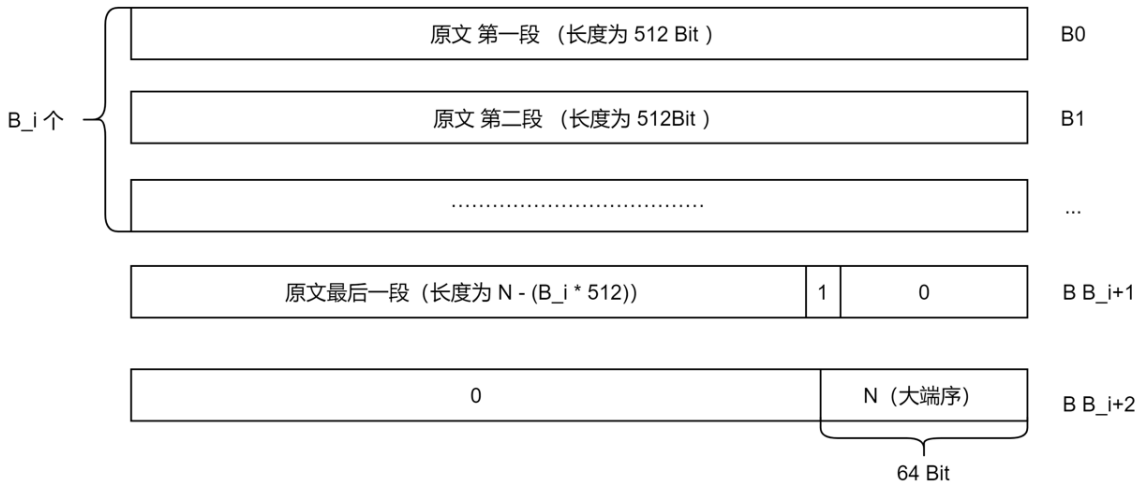


N > i * 512 Bit (i∈N* && i > 1):

512-L>64+1:



512-L<64+1:



第二步

将经过第一步的 M 叫为 Mc，将Mc分为 i 个 512 Bit 的B（上图中的 B0,B1,...,Bi-1）

对每一个B 计算 对应的W (64个)，参考代码：`BOOLEAN Mc_To_W(PSHA256_Data Data);`

t<16时：

W0~W15为B的划分(B为512Bit长，划分一个W 32 Bit长)

t>=16 && t<=63时：

$$W_t = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}$$

我的代码中，对于一个消息 Mc，对应的W的样式如下：每个B有63个W

	W0	W1	W2	W3	.....	W63
B0	nW[0] [0]					
B1		nW[1] [1]				
B2			nW[2] [2]			
...						
Bi-1	nW[i-1] [0]	nW[i-1] [1]	nW[i-1] [2]	nW[i-1] [3]		nW[i-1] [63]

算出每个B的W0~W63,填入上述表格中。

补充细节：

```
ULONG32 SR(ULONG32 x, int n) {//右循环n位
    ULONG32 y = x >> n;
    x = x << (32 - n);
    return x | y;
}

ULONG32 SIG0(ULONG32 x) {
    return SR(x, 7) ^ SR(x, 18) ^ ((x) >> 3);
}

ULONG32 SIG1(ULONG32 x) {
    return SR(x, 17) ^ SR(x, 19) ^ ((x) >> 10);
}
```

第三步

对于i个B（每个B有63个W），都要计算：

```
for (int i = 0; i < B_i; i++) {//每个B都要计算一次，共B_i个B
        A = nH[i][0];B = nH[i][1];C= nH[i][2];D= nH[i][3];E= nH[i][4];F= nH[i]
[5];G= nH[i][6];H= nH[i][7];
        for (int j = 0; j < 64; j++) {//每个B进入这个循环算出nH[i+1]
            T1 = H + Ep1(E) + Ch(E, F, G) + K[j] + Data->nW[i][j];
            T2 = Ep0(A) + Maj(A, B, C);
            H = G;
            G = F;
            F = E;
            E = D + T1;
            D = C;
            C = B;
            B = A;
            A = T1 + T2;
        }
        nH[i + 1][0] = A + nH[i][0];
        nH[i + 1][1] = B + nH[i][1];
        nH[i + 1][2] = C + nH[i][2];
        nH[i + 1][3] = D + nH[i][3];
        nH[i + 1][4] = E + nH[i][4];
        nH[i + 1][5] = F + nH[i][5];
        nH[i + 1][6] = G + nH[i][6];
        nH[i + 1][7] = H + nH[i][7];
    }
```

nH的结构如下：

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
nH[0]	0x6a09e667	0xbb67ae85	0x3c6ef372	0xa54ff53a	0x510e527f	0x9b05688c	0x1f83d9ab	0x5be0cd19
nH[1]		nH[1] [1]						
...								
nH[i-1]							nH[i-1] [6]	
nH[i]	nH[i] [0]	nH[i] [1]	nH[i] [2]	nH[i] [3]	nH[i] [4]	nH[i] [5]	nH[i] [6]	nH[i] [7]

共 i+1个nH（i为B的个数），每个nH有8个ULONG32。

nH[0]有初始值,如上表格所示。

最后得到的nH[i]即为SHA256最后结果。

补充细节：

```
ULONG32 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,0x84c87814,0x8cc70208,0x90befffa,0xa4506ceb,0xbef9a3f7,0xc67178f2
};

ULONG32 SL(ULONG32 x, int n) {//左循环n位
    ULONG32 y = x << n;
    x = x >> (32 - n);
    return x | y;
}

ULONG32 SR(ULONG32 x, int n) {//右循环n位
    ULONG32 y = x >> n;
    x = x << (32 - n);
    return x | y;
}

ULONG32 Ch(ULONG32 x, ULONG32 y, ULONG32 z) {
    return (x & y) ^ (~x & z);
}

ULONG32 Maj(ULONG32 x, ULONG32 y, ULONG32 z) {
    return ((x) & (y)) ^ ((x) & (z)) ^ ((y) & (z));
}

ULONG32 Ep0(ULONG32 x) {
```

```
    return SR(x, 2) ^ SR(x, 13) ^ SR(x, 22);
}

ULONG32 Ep1(ULONG32 x) {
    return SR(x, 6) ^ SR(x, 11) ^ SR(x, 25);
}
```