

SHA512

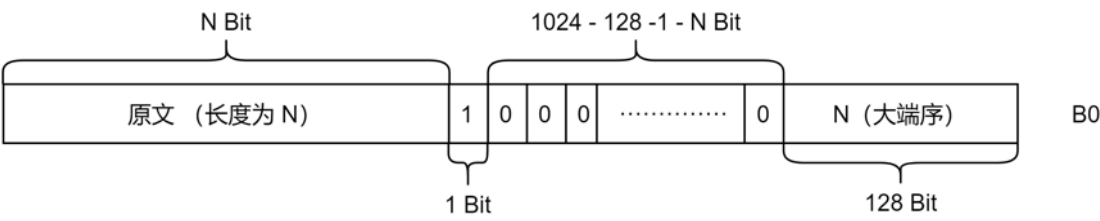
第一步

(与SHA256类似，只是B长度变为了1024，最后补充的长度N为128 bit)

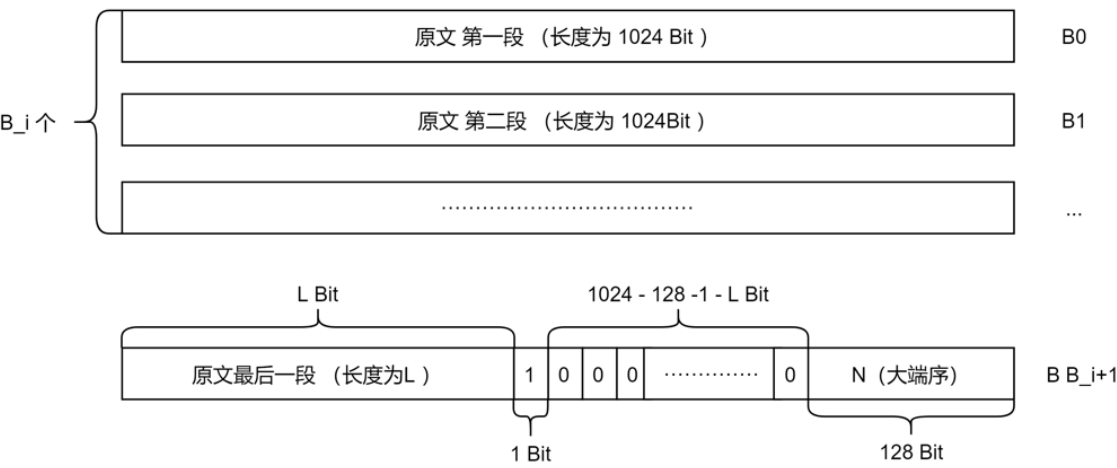
将需要散列的 16进制 字符串补充到 长度 $\text{mod}1024 = 0$ ，参考代码：SHA512_Data 结构 的 构造函数。

补充方式

设 需要散列的 文本为"M"，M的长度为N Bit， $B_i = N/1024$ ， $L = N\%1024$ 。

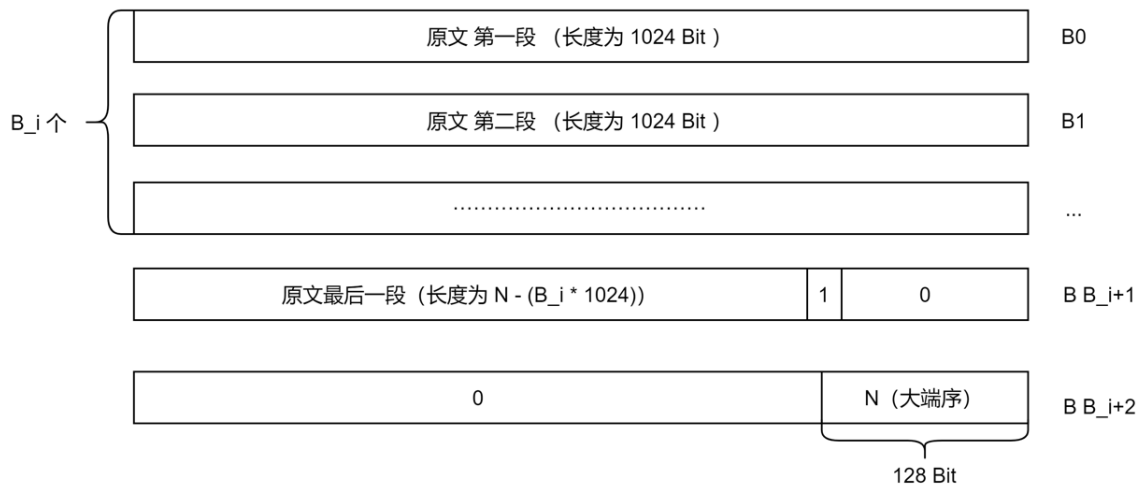


$N \leq 1024 - 128 - 1$ Bit:



$N > i * 1024 \text{Bit}$ ($i \in \mathbb{N}^*$ && $i > 1$):

$1024 - L > 128 + 1$:



第二步

将经过第一步的 M 叫为 M_c , 将 M_c 分为 i 个 1024Bit 的 B (上图中的 B_0, B_1, \dots, B_{i-1})

对每一个 B 计算 对应的 W (80个), 参考代码: `BOOLEAN MC_To_W(PSHA512_Data Data);`

$t < 16$ 时:

$W_0 \sim W_{15}$ 为 B 的划分 (B 为 1024Bit 长, 划分一个 W 64 Bit 长, 然后以八字节为单位转换端序)

$t \geq 16$ && $t \leq 79$ 时:

```
nW[i][j] = GAMMA1(nW[i][j - 2]) + nW[i][j - 7] + GAMMA0(nW[i][j - 15]) + nW[i][j - 16];
```

我的代码中, 对于一个消息 M_c , 对应的 W 的样式如下:

	W0	W1	W2	W3	W79
B0	nW[0][0]					
B1		nW[1][1]				
B2			nW[2][2]			
...						
B_{i-1}	nW[i-1][0]	nW[i-1][1]	nW[i-1][2]	nW[i-1][3]		nW[i-1][79]

算出每个 B 的 $W_0 \sim W_{79}$, 填入上述表格中。

补充细节:

```
#define GAMMA0(x) (ROR(x, 1) ^ ROR(x, 8) ^ LSR(x, 7))
#define GAMMA1(x) (ROR(x, 19) ^ ROR(x, 61) ^ LSR(x, 6))
```

第三步

对于i个B（每个B有80个W），都要计算：

```
for (ULONG64 i = 0; i < B_i; i++) {
    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 (ULONG64 j = 0; j < 80; j++) {
        T1 = H + SIGMA1(E) + CH(E, F, G) + K[j] + Data->nw[i][j];
        T2 = SIGMA0(A) + MA(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]	0x6A09E667F3BCC908ULL	0xBB67AE8584CAA73BULL	0x3C6EF372FE94F82BULL	0xA54FF53A5F1D36F1ULL	0x510E527FADE682D1ULL	0x9B05688C2B3E6C1FULL	0x1F83D9ABFB41BD68ULL	0x5BE0CD19137E2179ULL
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ULONG64。

nH[0]有初始值,如上表格所示。

最后得到的nH[i]即为SHA512最后结果。

对上述算法细节补充

```
#define LSR(x,n) (x >> n)
#define ROR(x,n) (LSR(x,n) | (x << (64 - n)))
#define MA(x,y,z) ((x & y) | (z & (x | y)))
#define CH(x,y,z) (z ^ (x & (y ^ z)))
#define GAMMA0(x) (ROR(x, 1) ^ ROR(x, 8) ^ LSR(x, 7))
#define GAMMA1(x) (ROR(x,19) ^ ROR(x,61) ^ LSR(x, 6))
#define SIGMA0(x) (ROR(x,28) ^ ROR(x,34) ^ ROR(x,39))
#define SIGMA1(x) (ROR(x,14) ^ ROR(x,18) ^ ROR(x,41))
```