SHA—1

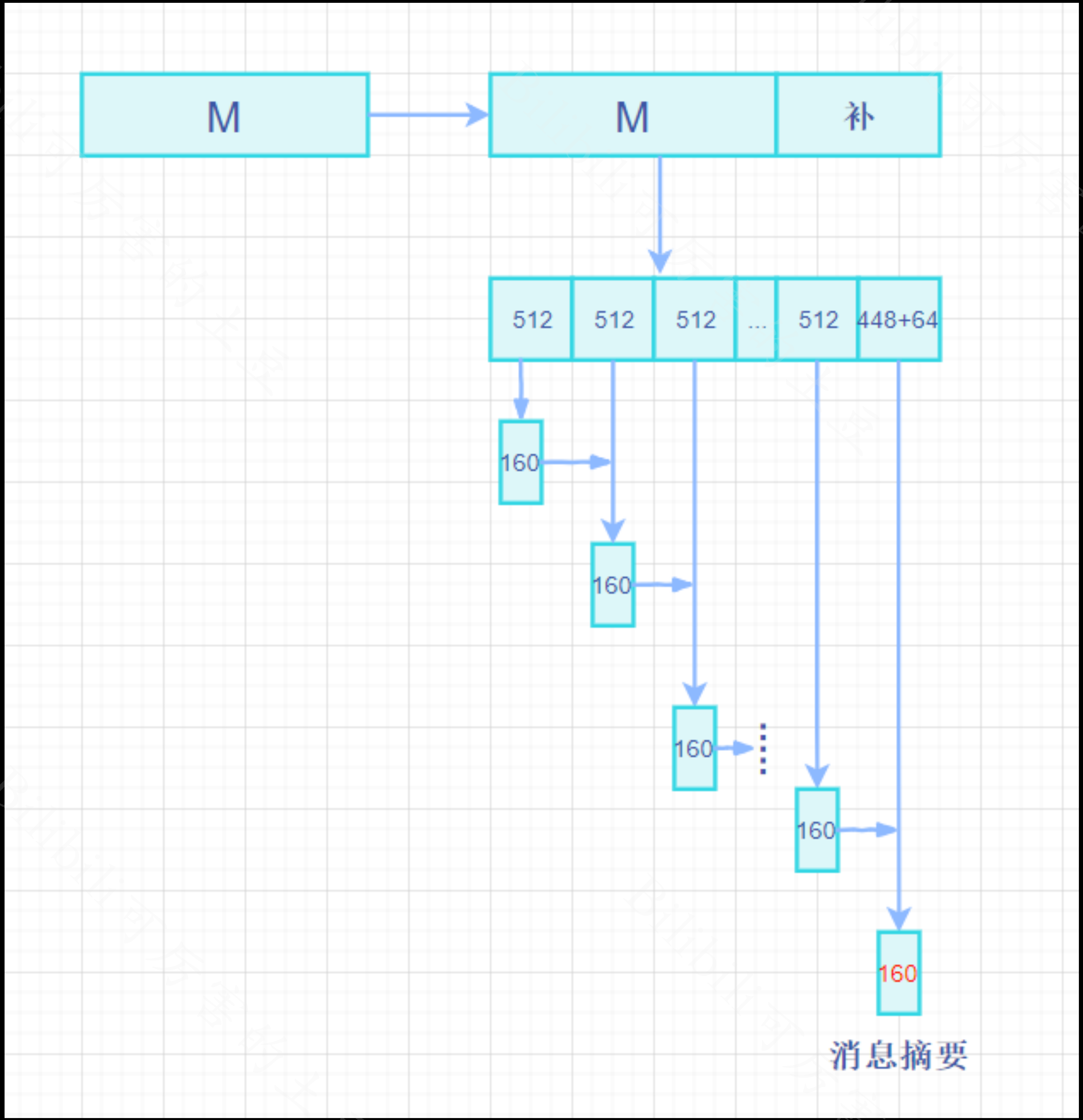
Secure Hash Algorithm

可厉害的土豆

SHA-1

输入：$0<L<2^{64}$

输出：160bit的消息摘要



- 补位
- 每个512bit的运算

补位

———怎么补

第一位：补1

其余位：补足够位数的0

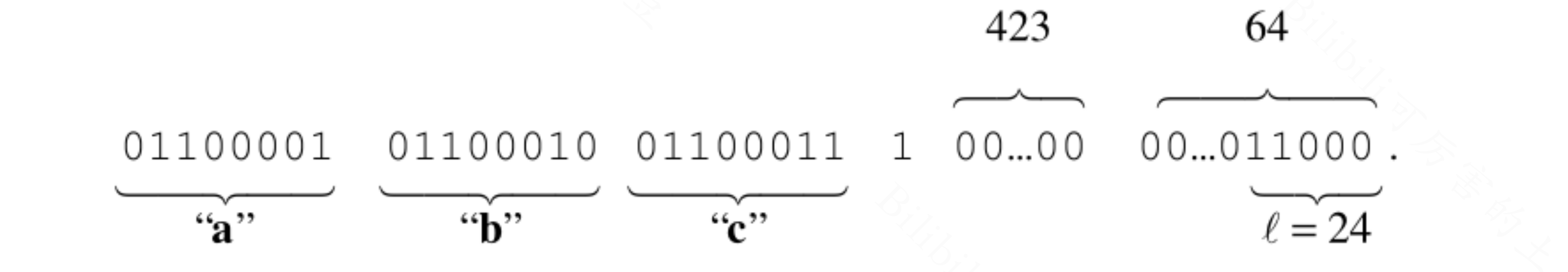
直至满足　L mod 512 = 448

512 - 448 = 64

剩余64位为消息的长度

补位

———举例



$$\underbrace{01100001}_{\text{“a”}}\quad\underbrace{01100010}_{\text{“b”}}\quad\underbrace{01100011}_{\text{“c”}}\quad 1\quad\overbrace{00\ldots00}^{423}\quad\overbrace{00\ldots0\underbrace{11000}_{\ell=24}}^{64}.$$

每个512bit的运算

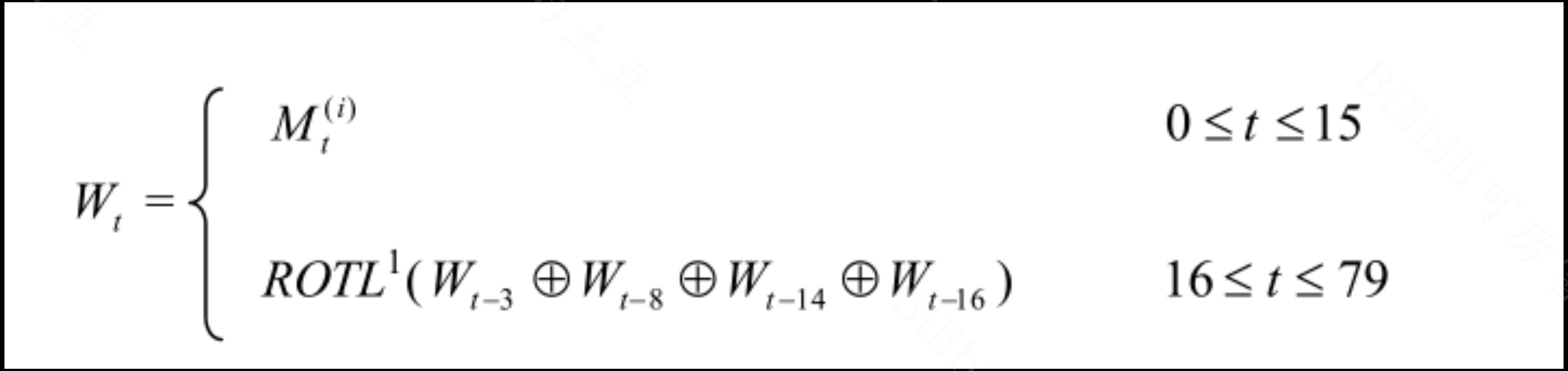
512 bit = 16份 * 32bit　　M[0], M[1]……M[15]

↓ 扩充

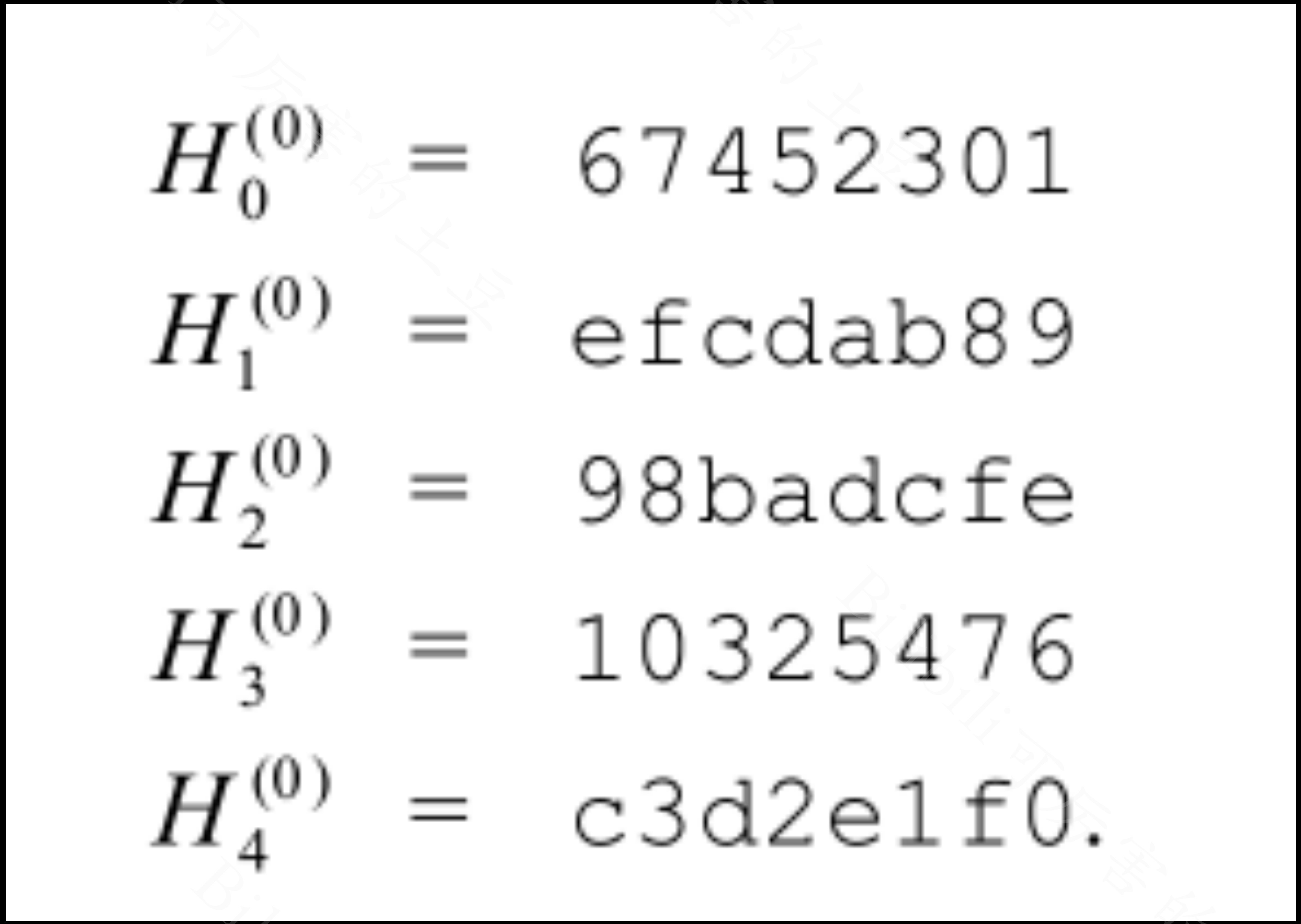
80份 * 32bit　　W[0], W[1]……W[79]

每个512bit的运算

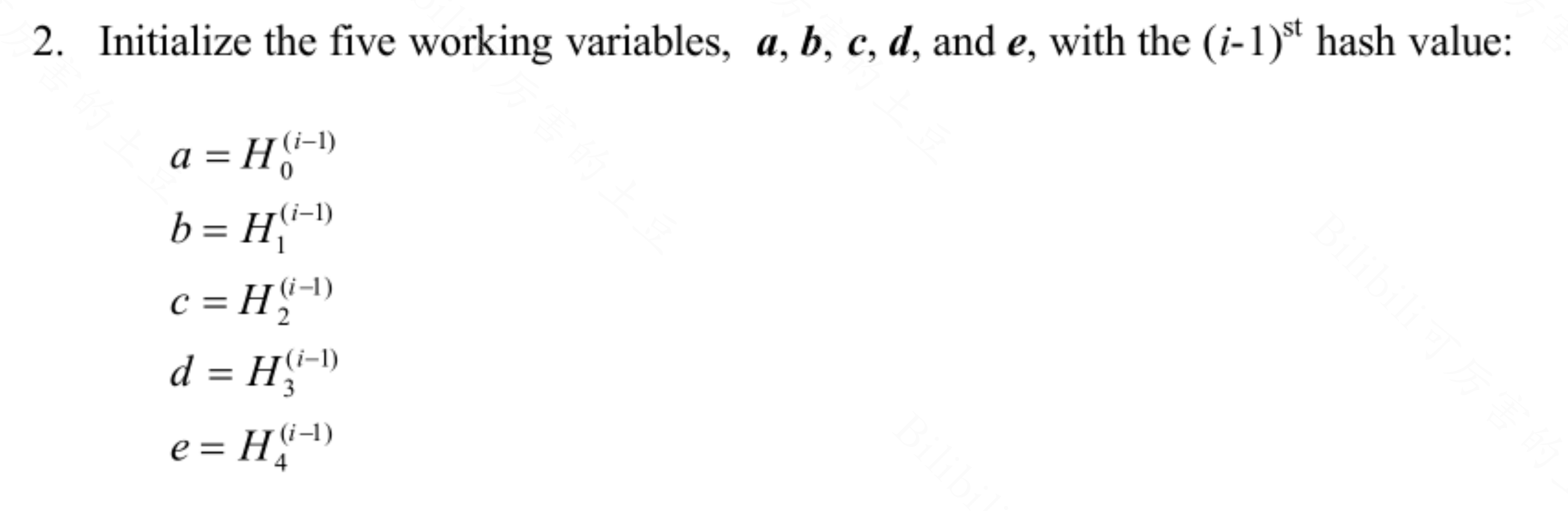
———预处理



$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \le t \le 79 \end{cases}$$

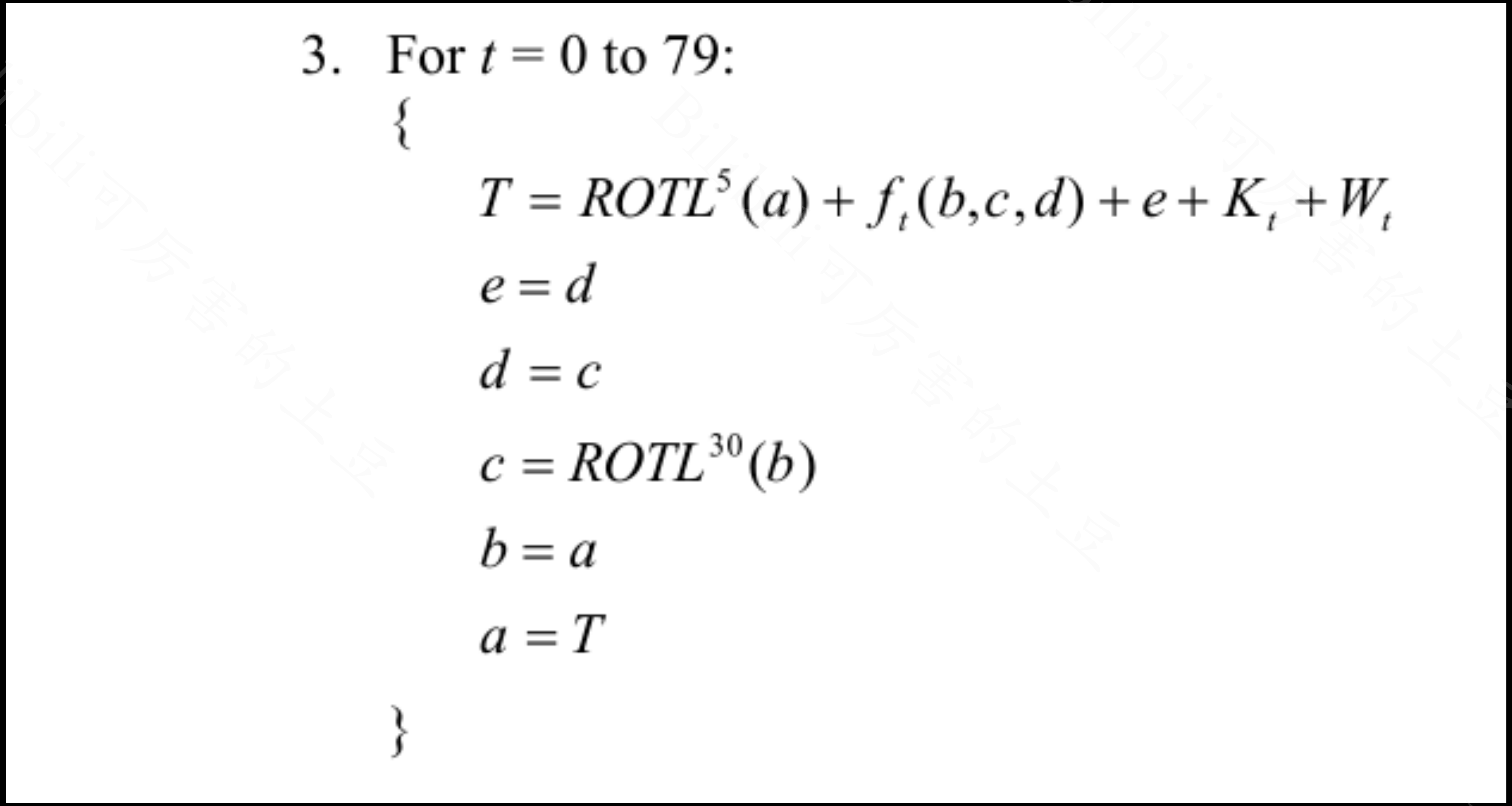


$$
\begin{aligned}
H_0^{(0)} &= \texttt{67452301} \\
H_1^{(0)} &= \texttt{efcdab89} \\
H_2^{(0)} &= \texttt{98badcfe} \\
H_3^{(0)} &= \texttt{10325476} \\
H_4^{(0)} &= \texttt{c3d2e1f0}.
\end{aligned}
$$

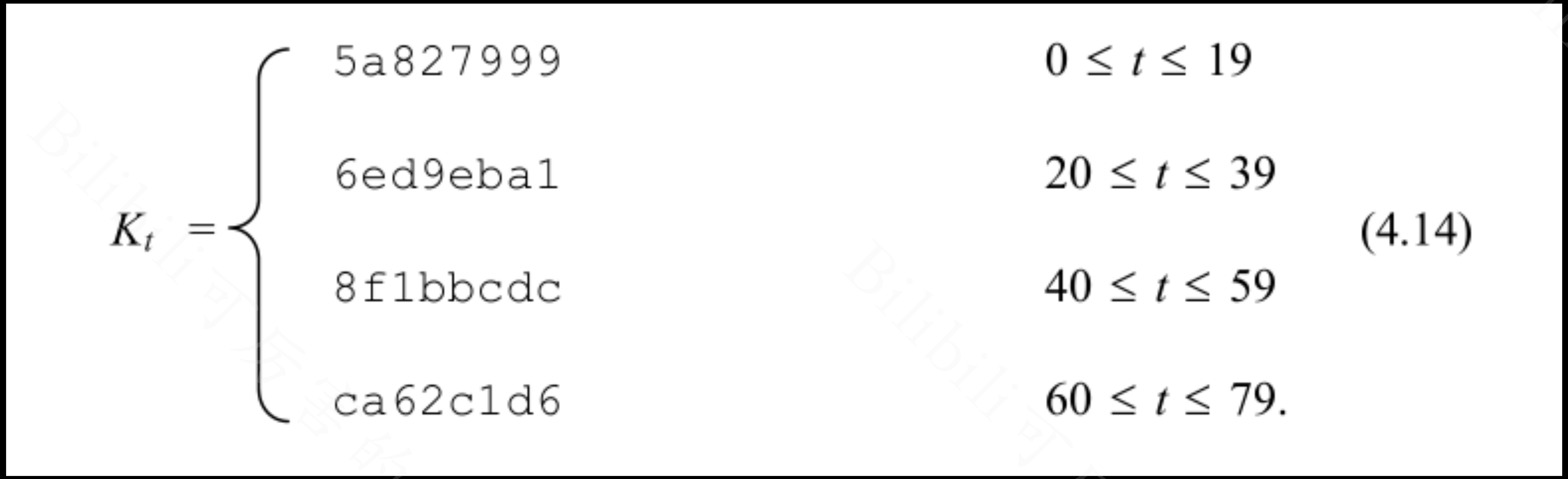


2. Initialize the five working variables, ***a***, ***b***, ***c***, ***d***, and ***e***, with the (*i*-1)$^{\text{st}}$ hash value:

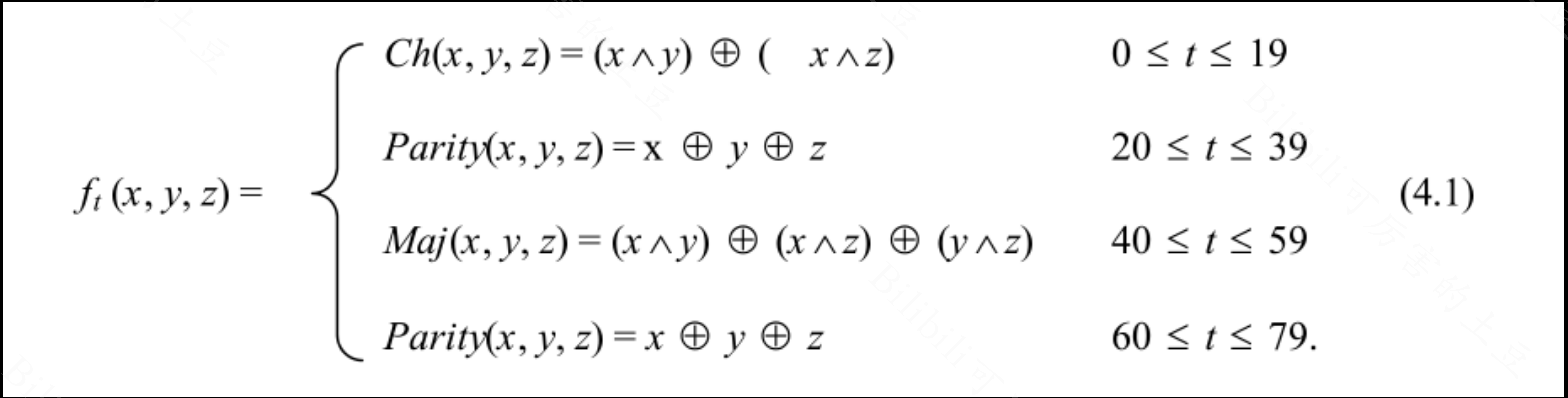
$$a = H_0^{(i-1)}$$
$$b = H_1^{(i-1)}$$
$$c = H_2^{(i-1)}$$
$$d = H_3^{(i-1)}$$
$$e = H_4^{(i-1)}$$



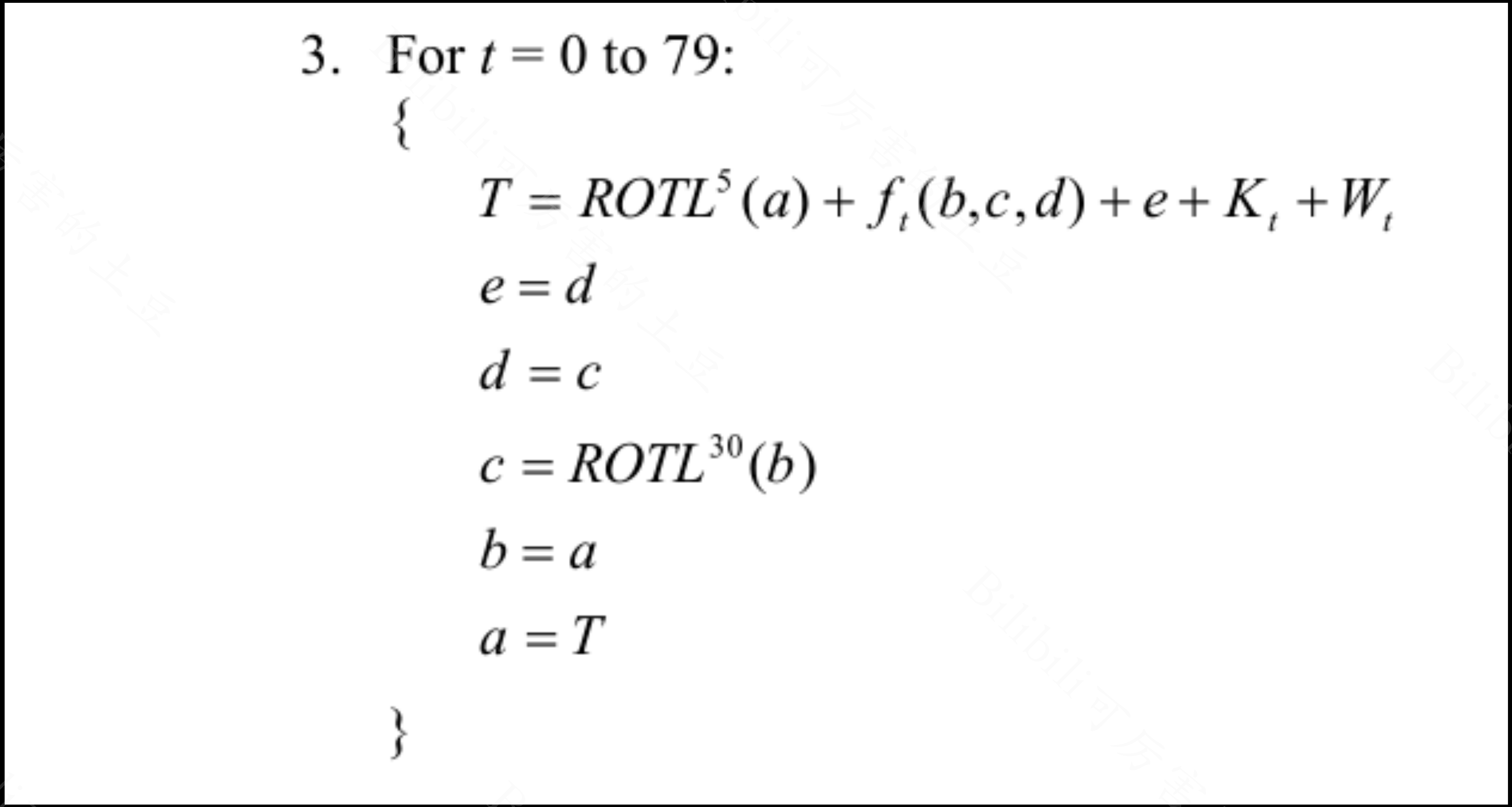
```
3.  For $t = 0$ to 79:
    {
        $T = ROTL^{5}(a) + f_{t}(b,c,d) + e + K_{t} + W_{t}$
        $e = d$
        $d = c$
        $c = ROTL^{30}(b)$
        $b = a$
        $a = T$
    }
```



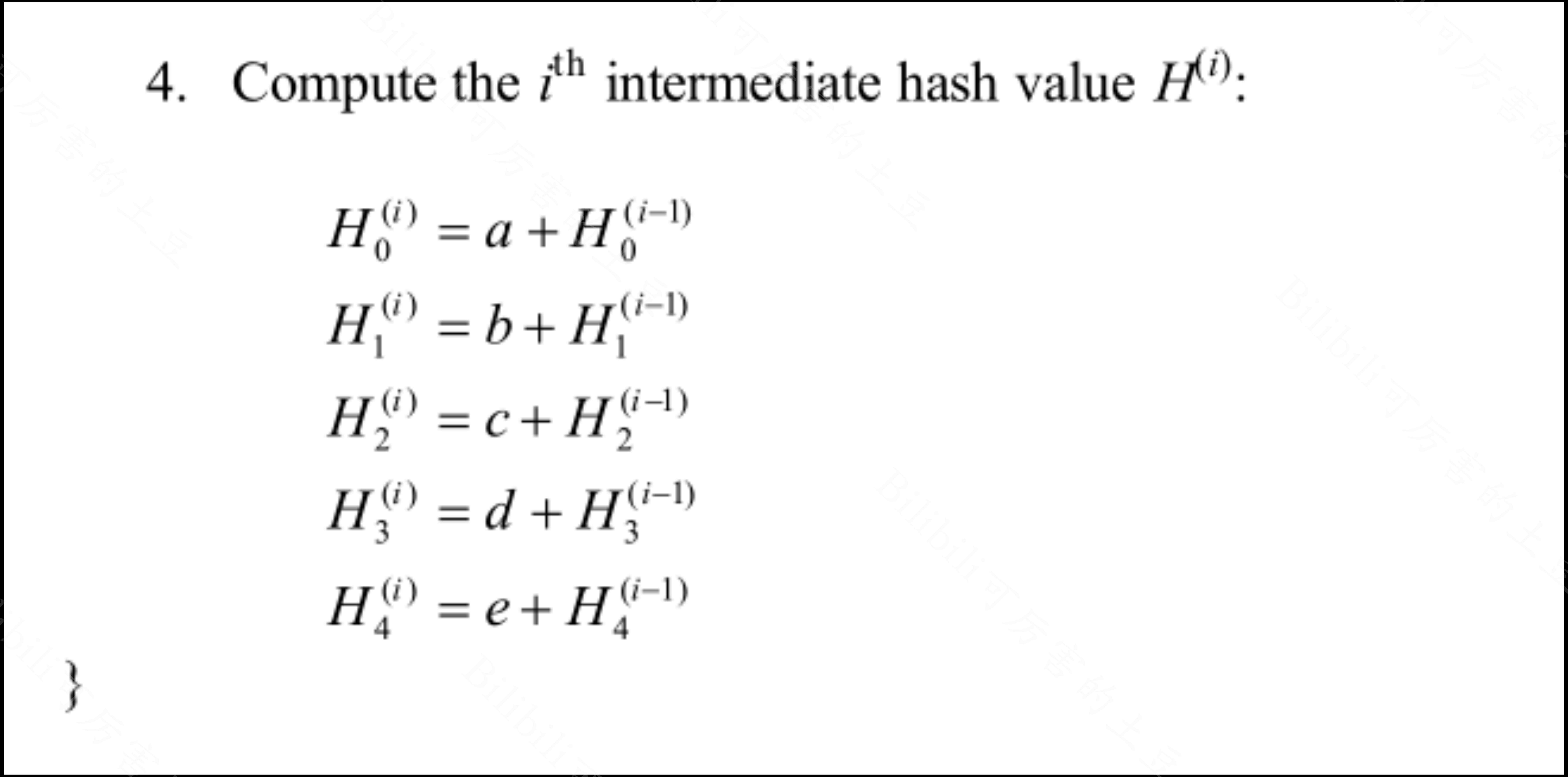
$$K_t = \begin{cases} \texttt{5a827999} & 0 \le t \le 19 \\ \texttt{6ed9eba1} & 20 \le t \le 39 \\ \texttt{8f1bbcdc} & 40 \le t \le 59 \\ \texttt{ca62c1d6} & 60 \le t \le 79. \end{cases} \qquad (4.14)$$



$$f_t(x, y, z) = \begin{cases} Ch(x, y, z) = (x \wedge y) \oplus (\quad x \wedge z) & 0 \le t \le 19 \\ Parity(x, y, z) = \mathrm{x} \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x, y, z) = x \oplus y \oplus z & 60 \le t \le 79. \end{cases} \qquad (4.1)$$



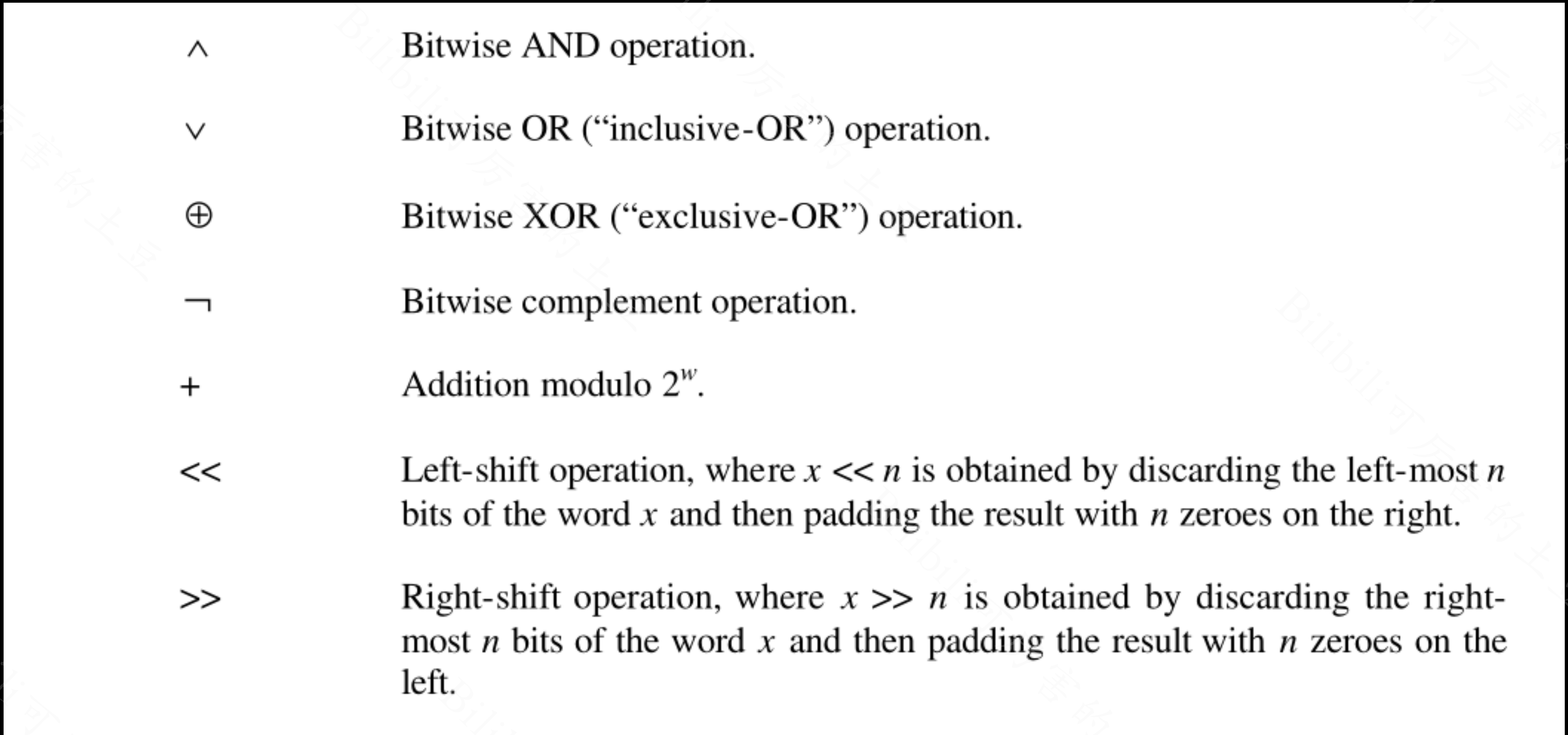
```
3.  For $t = 0$ to 79:
    {
        $T = ROTL^{5}(a) + f_{t}(b,c,d) + e + K_{t} + W_{t}$
        $e = d$
        $d = c$
        $c = ROTL^{30}(b)$
        $b = a$
        $a = T$
    }
```



4. Compute the i^{th} intermediate hash value $H^{(i)}$:

$$
\begin{aligned}
H_0^{(i)} &= a + H_0^{(i-1)} \\
H_1^{(i)} &= b + H_1^{(i-1)} \\
H_2^{(i)} &= c + H_2^{(i-1)} \\
H_3^{(i)} &= d + H_3^{(i-1)} \\
H_4^{(i)} &= e + H_4^{(i-1)}
\end{aligned}
$$

}



$\wedge$ Bitwise AND operation.

$\vee$ Bitwise OR (“inclusive-OR”) operation.

$\oplus$ Bitwise XOR (“exclusive-OR”) operation.

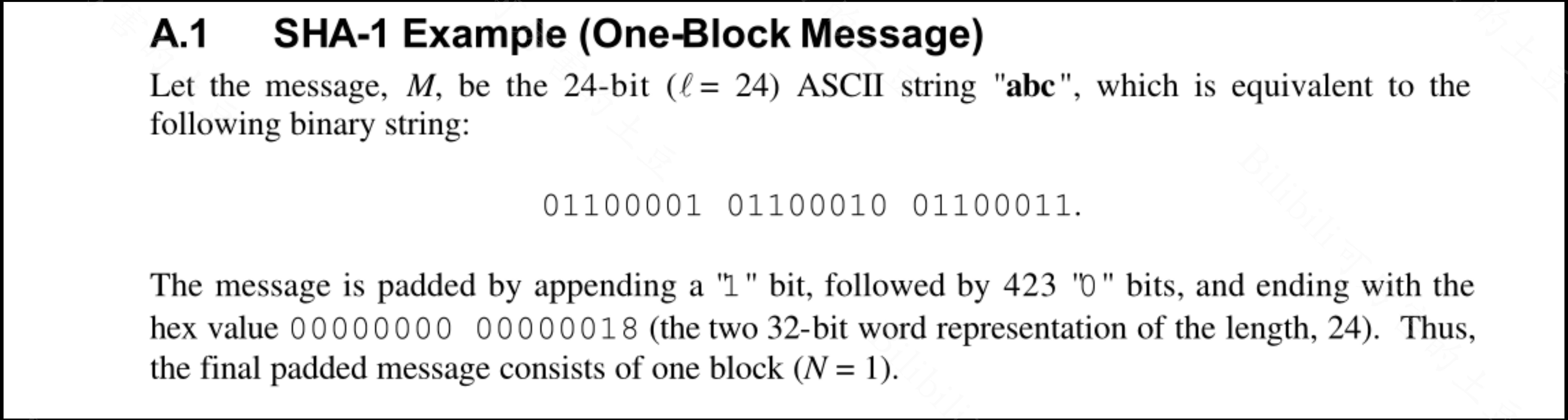
$\neg$ Bitwise complement operation.

$+$ Addition modulo 2^w.

$<<$ Left-shift operation, where $x << n$ is obtained by discarding the left-most n bits of the word x and then padding the result with n zeroes on the right.

$>>$ Right-shift operation, where $x >> n$ is obtained by discarding the right-most n bits of the word x and then padding the result with n zeroes on the left.

举个例子



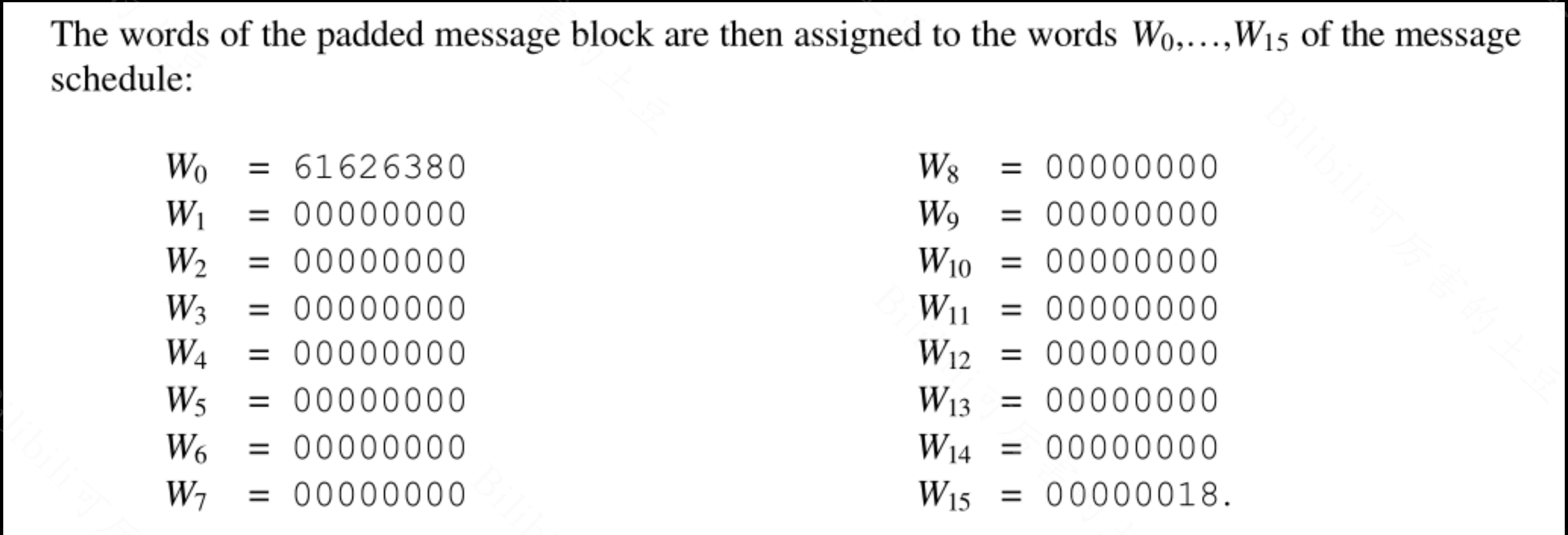
A.1 SHA-1 Example (One-Block Message)

Let the message, M, be the 24-bit ($\ell = 24$) ASCII string "**abc**", which is equivalent to the following binary string:

```
01100001 01100010 01100011.
```

The message is padded by appending a "`1`" bit, followed by 423 "`0`" bits, and ending with the hex value `00000000 00000018` (the two 32-bit word representation of the length, 24). Thus, the final padded message consists of one block ($N = 1$).

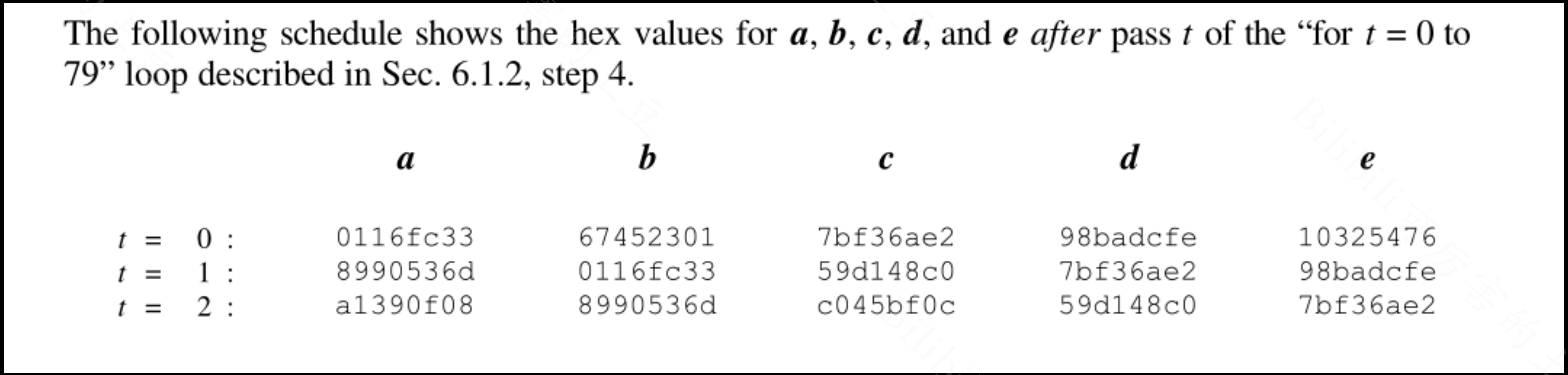
举个例子



The words of the padded message block are then assigned to the words $W_0,\ldots,W_{15}$ of the message schedule:

W_0	= 61626380	W_8	= 00000000
W_1	= 00000000	W_9	= 00000000
W_2	= 00000000	W_{10}	= 00000000
W_3	= 00000000	W_{11}	= 00000000
W_4	= 00000000	W_{12}	= 00000000
W_5	= 00000000	W_{13}	= 00000000
W_6	= 00000000	W_{14}	= 00000000
W_7	= 00000000	W_{15}	= 00000018.

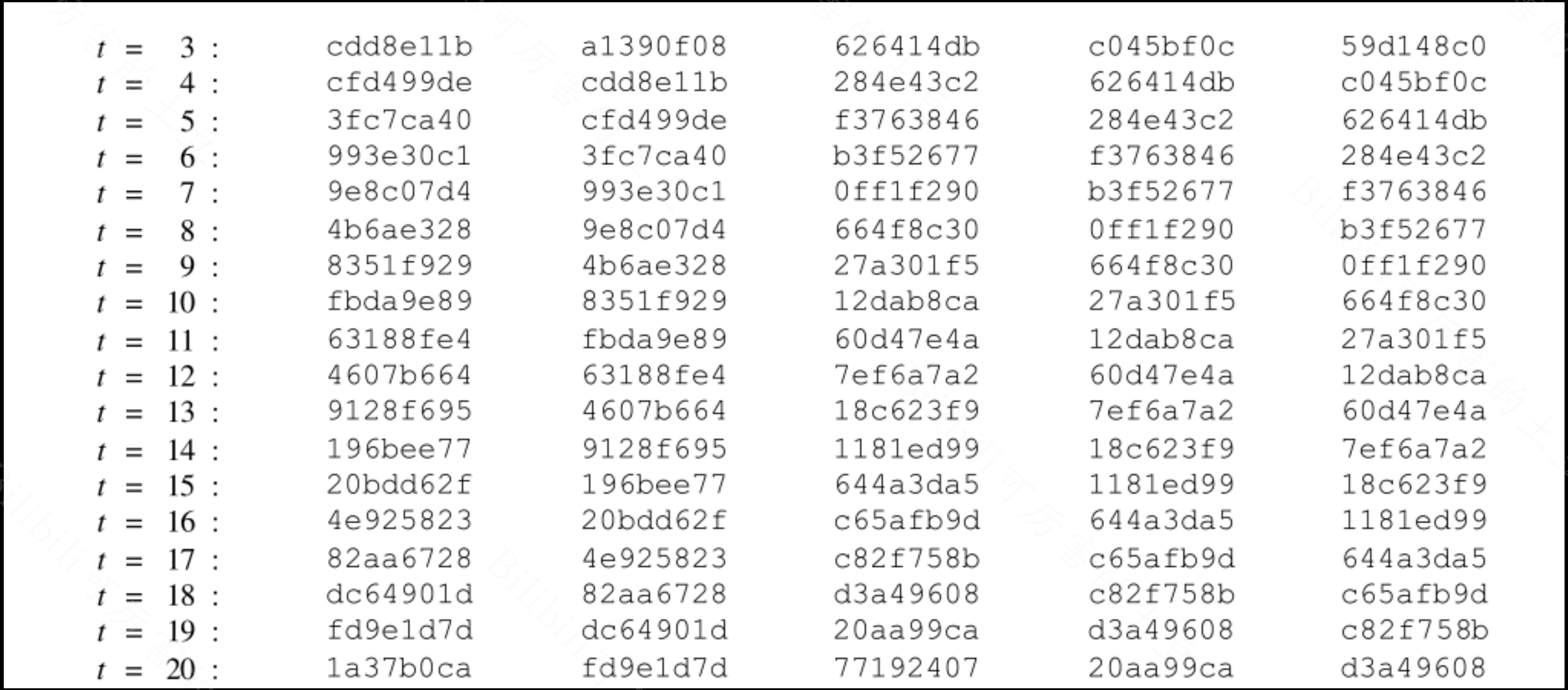
举个例子



The following schedule shows the hex values for ***a***, ***b***, ***c***, ***d***, and ***e*** *after* pass *t* of the "for *t* = 0 to 79" loop described in Sec. 6.1.2, step 4.

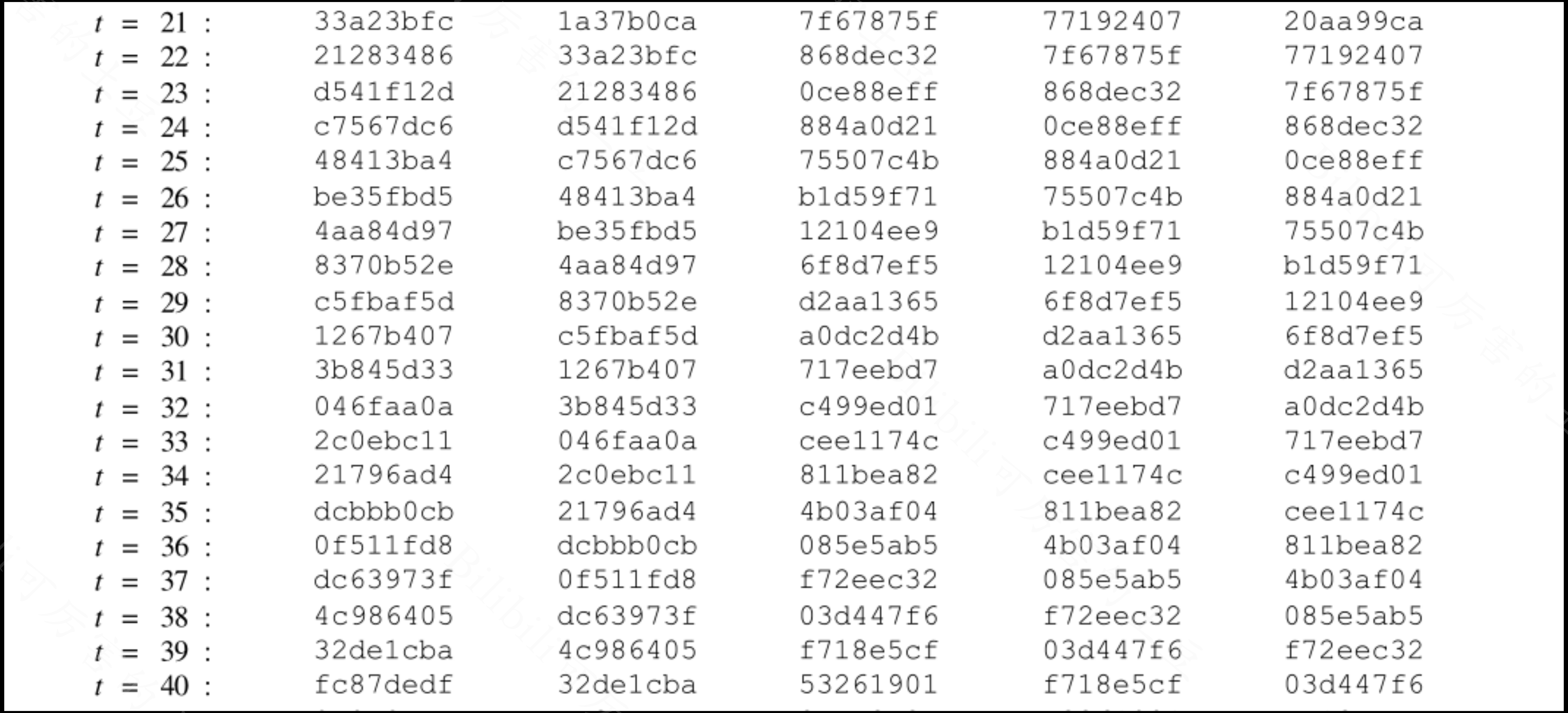
	a	***b***	***c***	***d***	***e***
t = 0 :	0116fc33	67452301	7bf36ae2	98badcfe	10325476
t = 1 :	8990536d	0116fc33	59d148c0	7bf36ae2	98badcfe
t = 2 :	a1390f08	8990536d	c045bf0c	59d148c0	7bf36ae2

举个例子



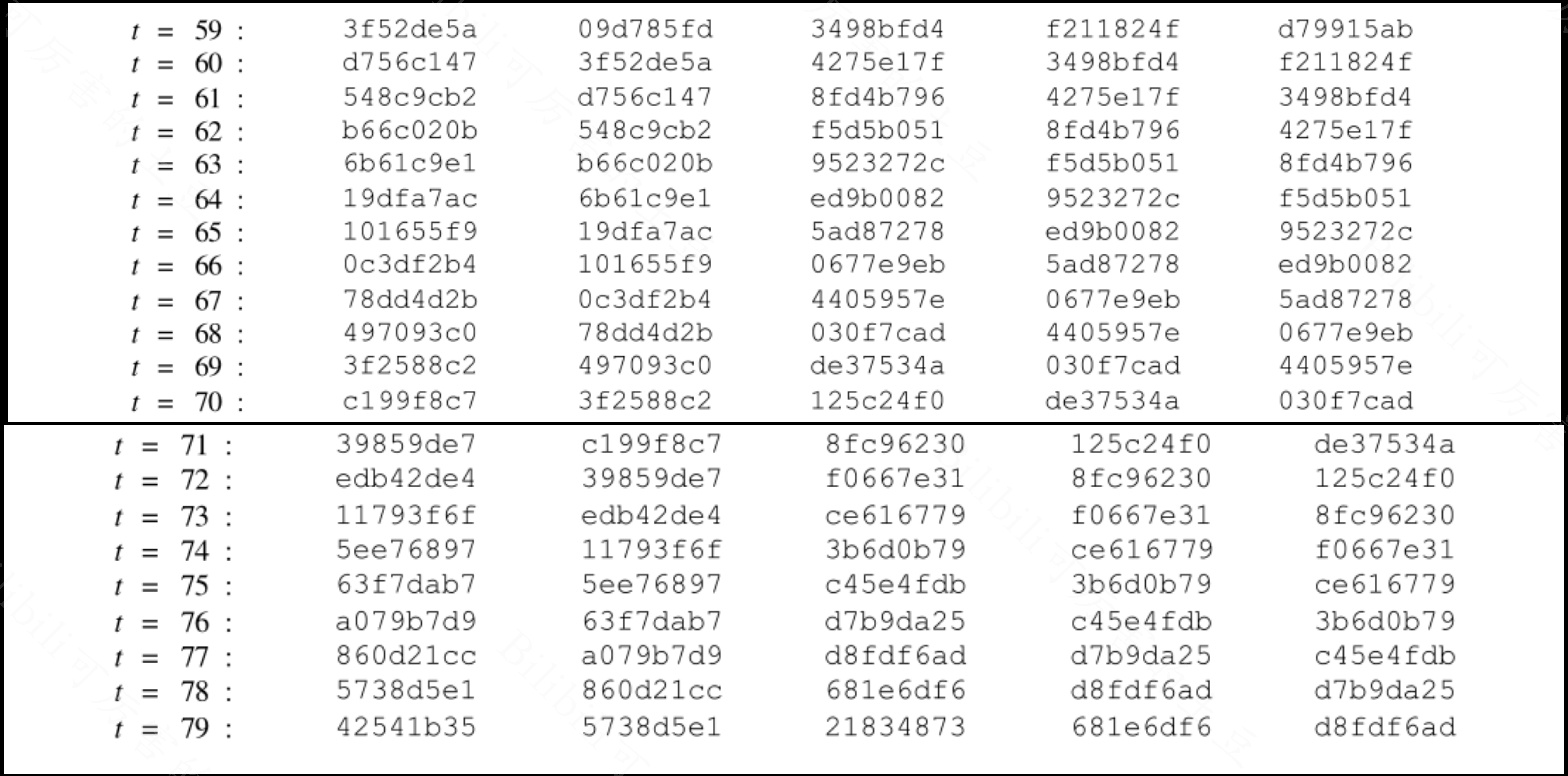
$t = 3$:	cdd8e11b	a1390f08	626414db	c045bf0c	59d148c0
$t = 4$:	cfd499de	cdd8e11b	284e43c2	626414db	c045bf0c
$t = 5$:	3fc7ca40	cfd499de	f3763846	284e43c2	626414db
$t = 6$:	993e30c1	3fc7ca40	b3f52677	f3763846	284e43c2
$t = 7$:	9e8c07d4	993e30c1	0ff1f290	b3f52677	f3763846
$t = 8$:	4b6ae328	9e8c07d4	664f8c30	0ff1f290	b3f52677
$t = 9$:	8351f929	4b6ae328	27a301f5	664f8c30	0ff1f290
$t = 10$:	fbda9e89	8351f929	12dab8ca	27a301f5	664f8c30
$t = 11$:	63188fe4	fbda9e89	60d47e4a	12dab8ca	27a301f5
$t = 12$:	4607b664	63188fe4	7ef6a7a2	60d47e4a	12dab8ca
$t = 13$:	9128f695	4607b664	18c623f9	7ef6a7a2	60d47e4a
$t = 14$:	196bee77	9128f695	1181ed99	18c623f9	7ef6a7a2
$t = 15$:	20bdd62f	196bee77	644a3da5	1181ed99	18c623f9
$t = 16$:	4e925823	20bdd62f	c65afb9d	644a3da5	1181ed99
$t = 17$:	82aa6728	4e925823	c82f758b	c65afb9d	644a3da5
$t = 18$:	dc64901d	82aa6728	d3a49608	c82f758b	c65afb9d
$t = 19$:	fd9e1d7d	dc64901d	20aa99ca	d3a49608	c82f758b
$t = 20$:	1a37b0ca	fd9e1d7d	77192407	20aa99ca	d3a49608

举个例子



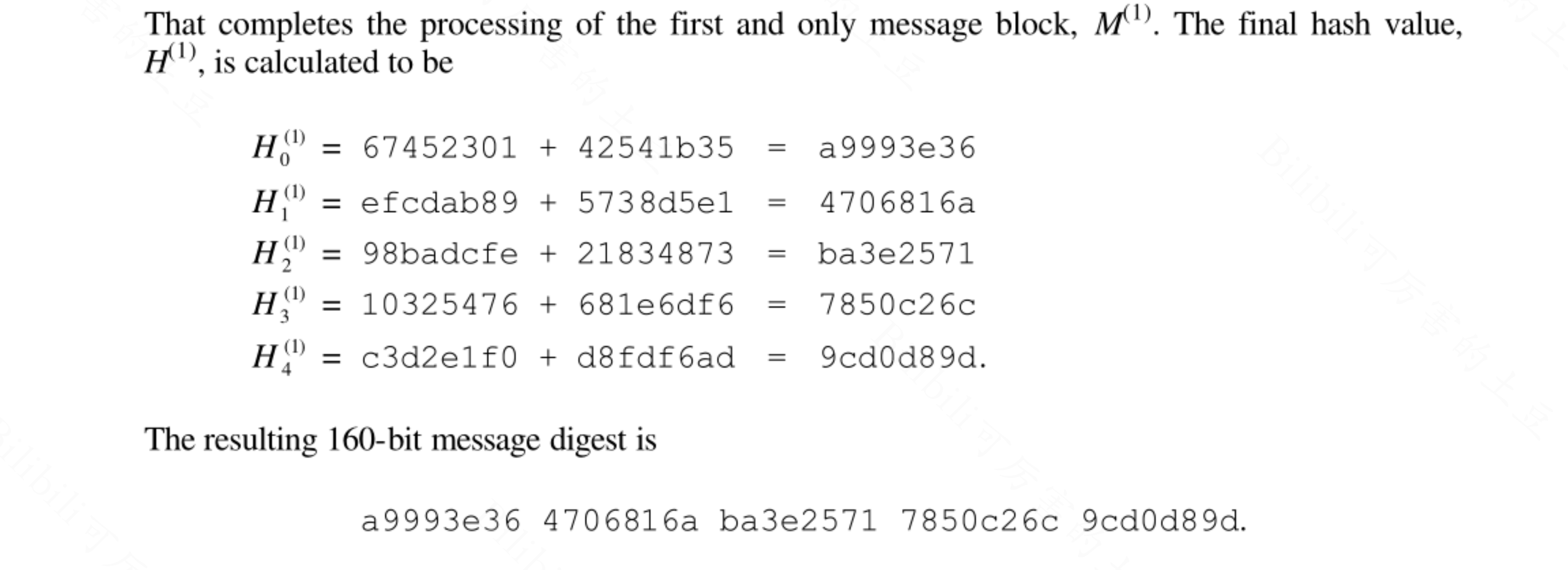
t = 21 :	33a23bfc	1a37b0ca	7f67875f	77192407	20aa99ca
t = 22 :	21283486	33a23bfc	868dec32	7f67875f	77192407
t = 23 :	d541f12d	21283486	0ce88eff	868dec32	7f67875f
t = 24 :	c7567dc6	d541f12d	884a0d21	0ce88eff	868dec32
t = 25 :	48413ba4	c7567dc6	75507c4b	884a0d21	0ce88eff
t = 26 :	be35fbd5	48413ba4	b1d59f71	75507c4b	884a0d21
t = 27 :	4aa84d97	be35fbd5	12104ee9	b1d59f71	75507c4b
t = 28 :	8370b52e	4aa84d97	6f8d7ef5	12104ee9	b1d59f71
t = 29 :	c5fbaf5d	8370b52e	d2aa1365	6f8d7ef5	12104ee9
t = 30 :	1267b407	c5fbaf5d	a0dc2d4b	d2aa1365	6f8d7ef5
t = 31 :	3b845d33	1267b407	717eebd7	a0dc2d4b	d2aa1365
t = 32 :	046faa0a	3b845d33	c499ed01	717eebd7	a0dc2d4b
t = 33 :	2c0ebc11	046faa0a	cee1174c	c499ed01	717eebd7
t = 34 :	21796ad4	2c0ebc11	811bea82	cee1174c	c499ed01
t = 35 :	dcbbb0cb	21796ad4	4b03af04	811bea82	cee1174c
t = 36 :	0f511fd8	dcbbb0cb	085e5ab5	4b03af04	811bea82
t = 37 :	dc63973f	0f511fd8	f72eec32	085e5ab5	4b03af04
t = 38 :	4c986405	dc63973f	03d447f6	f72eec32	085e5ab5
t = 39 :	32de1cba	4c986405	f718e5cf	03d447f6	f72eec32
t = 40 :	fc87dedf	32de1cba	53261901	f718e5cf	03d447f6

举个例子



t = 59 :	3f52de5a	09d785fd	3498bfd4	f211824f	d79915ab
t = 60 :	d756c147	3f52de5a	4275e17f	3498bfd4	f211824f
t = 61 :	548c9cb2	d756c147	8fd4b796	4275e17f	3498bfd4
t = 62 :	b66c020b	548c9cb2	f5d5b051	8fd4b796	4275e17f
t = 63 :	6b61c9e1	b66c020b	9523272c	f5d5b051	8fd4b796
t = 64 :	19dfa7ac	6b61c9e1	ed9b0082	9523272c	f5d5b051
t = 65 :	101655f9	19dfa7ac	5ad87278	ed9b0082	9523272c
t = 66 :	0c3df2b4	101655f9	0677e9eb	5ad87278	ed9b0082
t = 67 :	78dd4d2b	0c3df2b4	4405957e	0677e9eb	5ad87278
t = 68 :	497093c0	78dd4d2b	030f7cad	4405957e	0677e9eb
t = 69 :	3f2588c2	497093c0	de37534a	030f7cad	4405957e
t = 70 :	c199f8c7	3f2588c2	125c24f0	de37534a	030f7cad
t = 71 :	39859de7	c199f8c7	8fc96230	125c24f0	de37534a
t = 72 :	edb42de4	39859de7	f0667e31	8fc96230	125c24f0
t = 73 :	11793f6f	edb42de4	ce616779	f0667e31	8fc96230
t = 74 :	5ee76897	11793f6f	3b6d0b79	ce616779	f0667e31
t = 75 :	63f7dab7	5ee76897	c45e4fdb	3b6d0b79	ce616779
t = 76 :	a079b7d9	63f7dab7	d7b9da25	c45e4fdb	3b6d0b79
t = 77 :	860d21cc	a079b7d9	d8fdf6ad	d7b9da25	c45e4fdb
t = 78 :	5738d5e1	860d21cc	681e6df6	d8fdf6ad	d7b9da25
t = 79 :	42541b35	5738d5e1	21834873	681e6df6	d8fdf6ad

举个例子



That completes the processing of the first and only message block, $M^{(1)}$. The final hash value, $H^{(1)}$, is calculated to be

$$H_0^{(1)} = \texttt{67452301} + \texttt{42541b35} = \texttt{a9993e36}$$
$$H_1^{(1)} = \texttt{efcdab89} + \texttt{5738d5e1} = \texttt{4706816a}$$
$$H_2^{(1)} = \texttt{98badcfe} + \texttt{21834873} = \texttt{ba3e2571}$$
$$H_3^{(1)} = \texttt{10325476} + \texttt{681e6df6} = \texttt{7850c26c}$$
$$H_4^{(1)} = \texttt{c3d2e1f0} + \texttt{d8fdf6ad} = \texttt{9cd0d89d}.$$

The resulting 160-bit message digest is

```
a9993e36 4706816a ba3e2571 7850c26c 9cd0d89d.
```

参考资料

Secure Hash Standard: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

“

感谢观看
祝你每顿吃饱
每晚睡好
身体健康
学业有成
工作顺利

”

可厉害的土豆