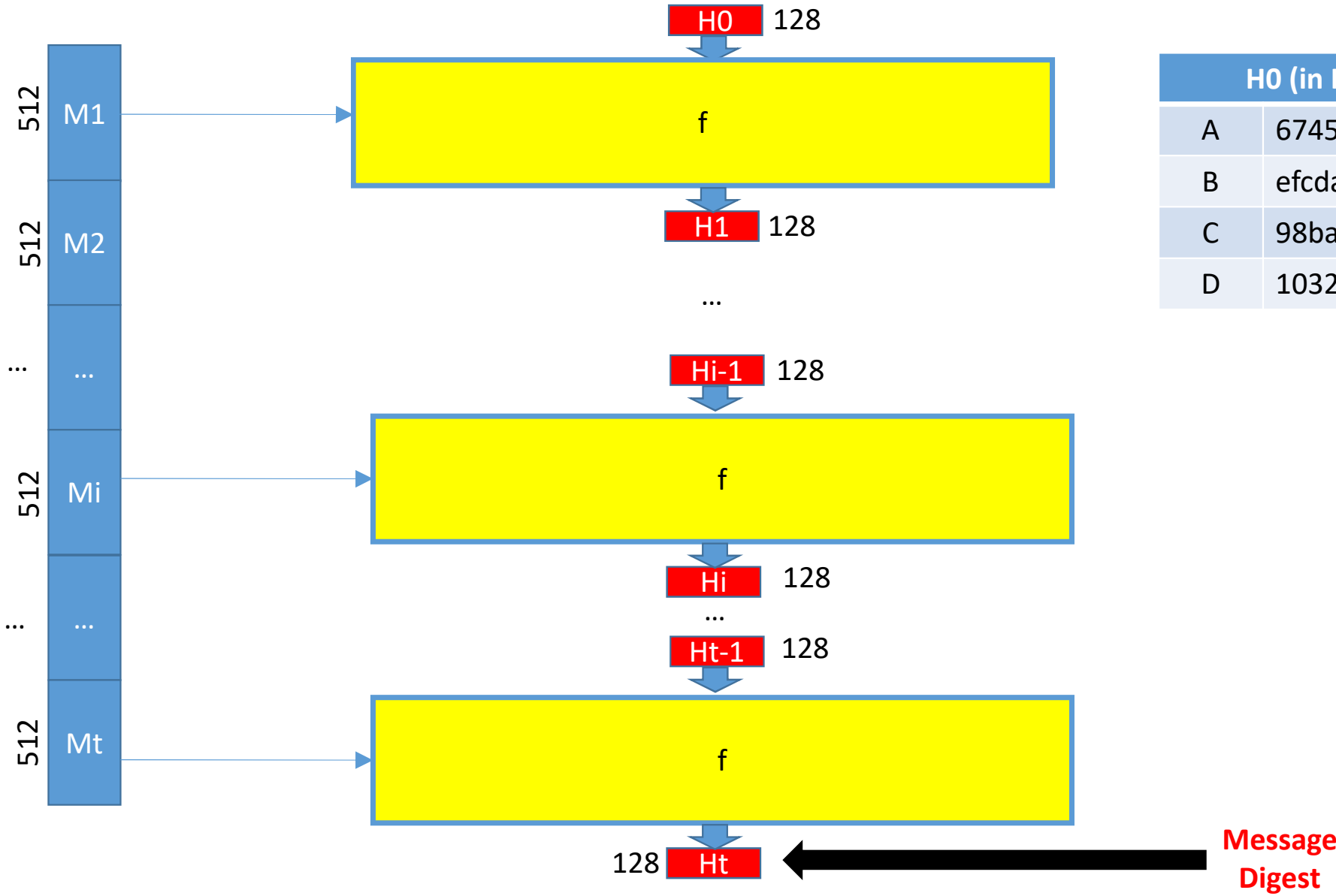
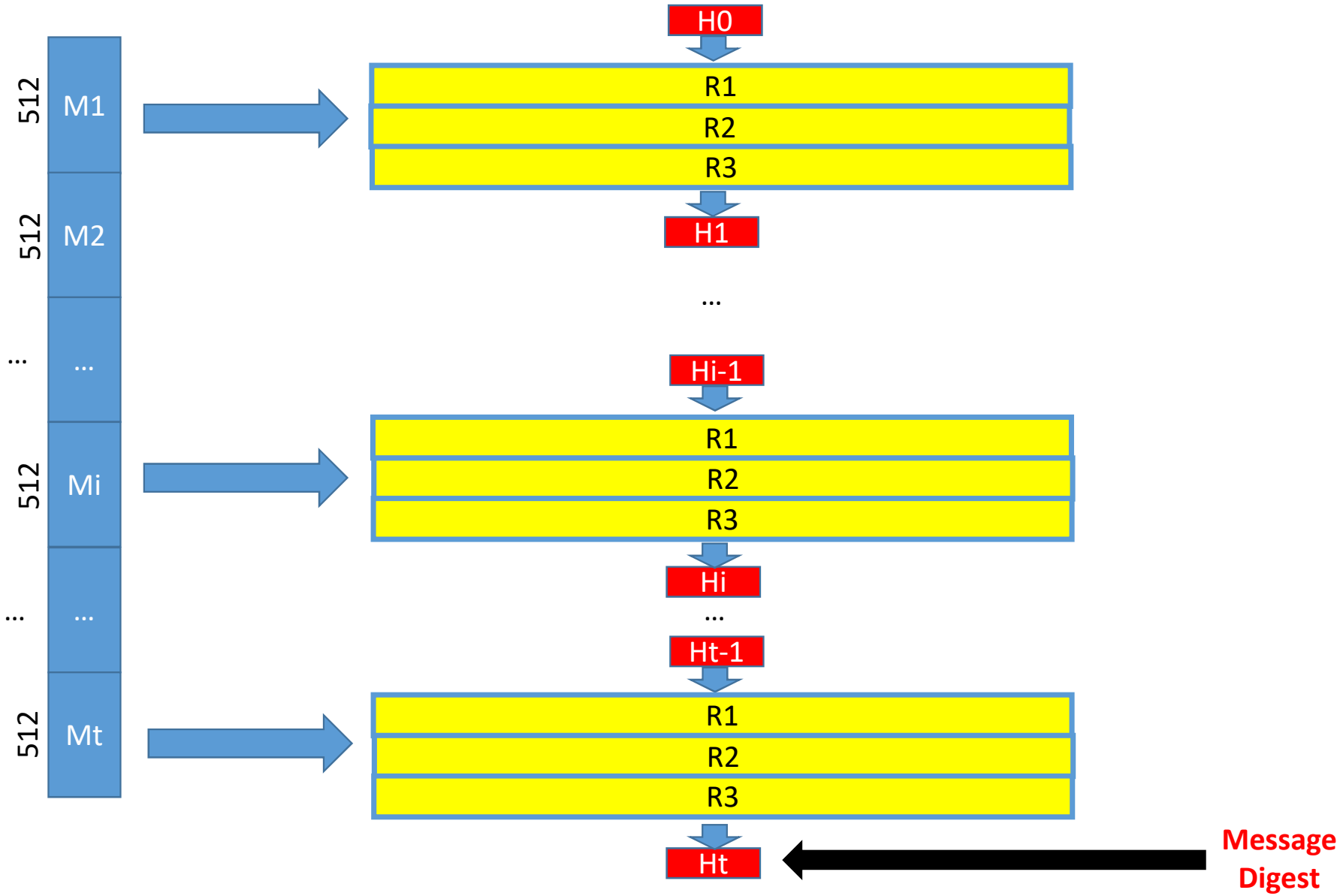
MD4

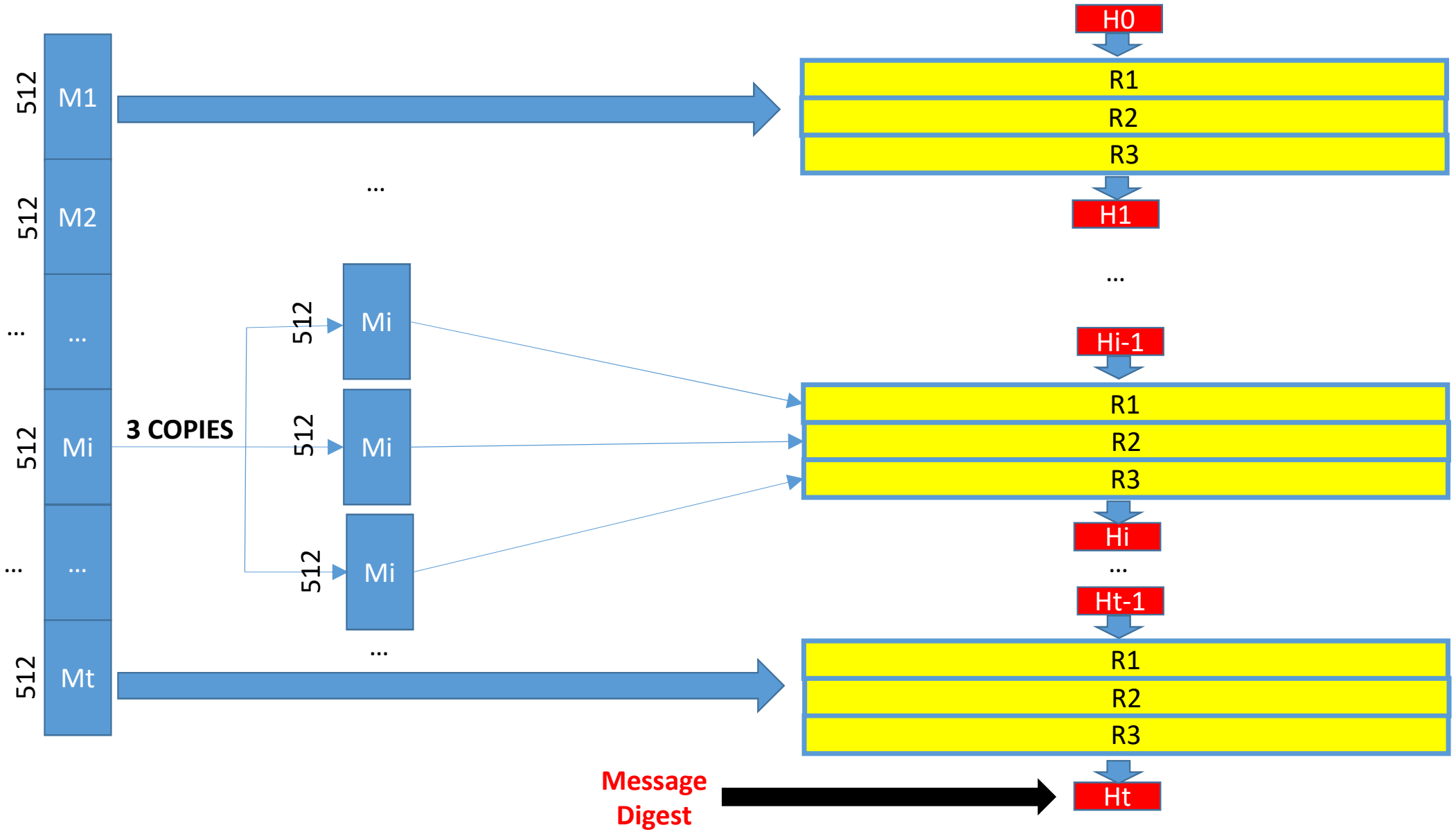
Dr. Ruchira Naskar

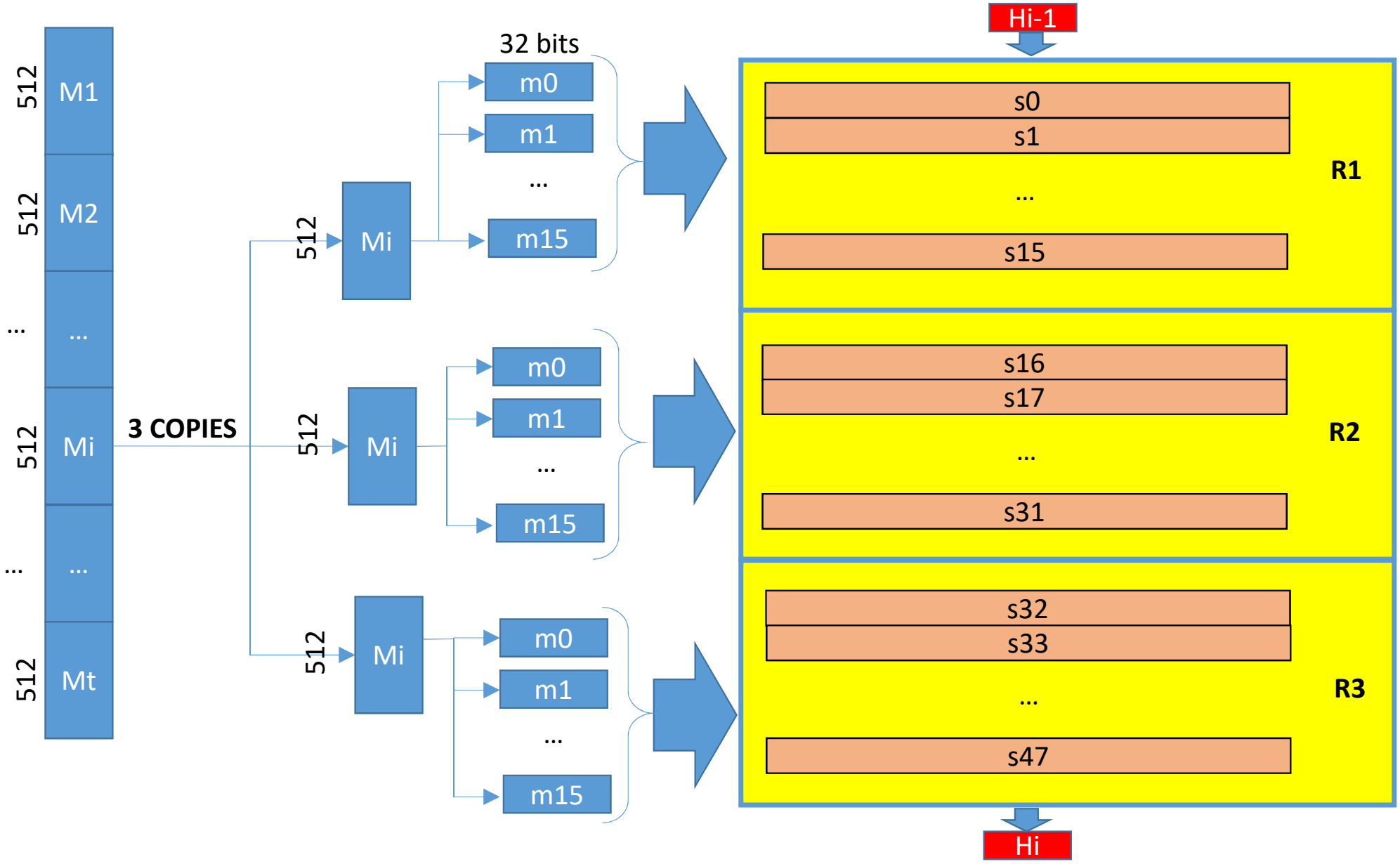
Padding in MD4

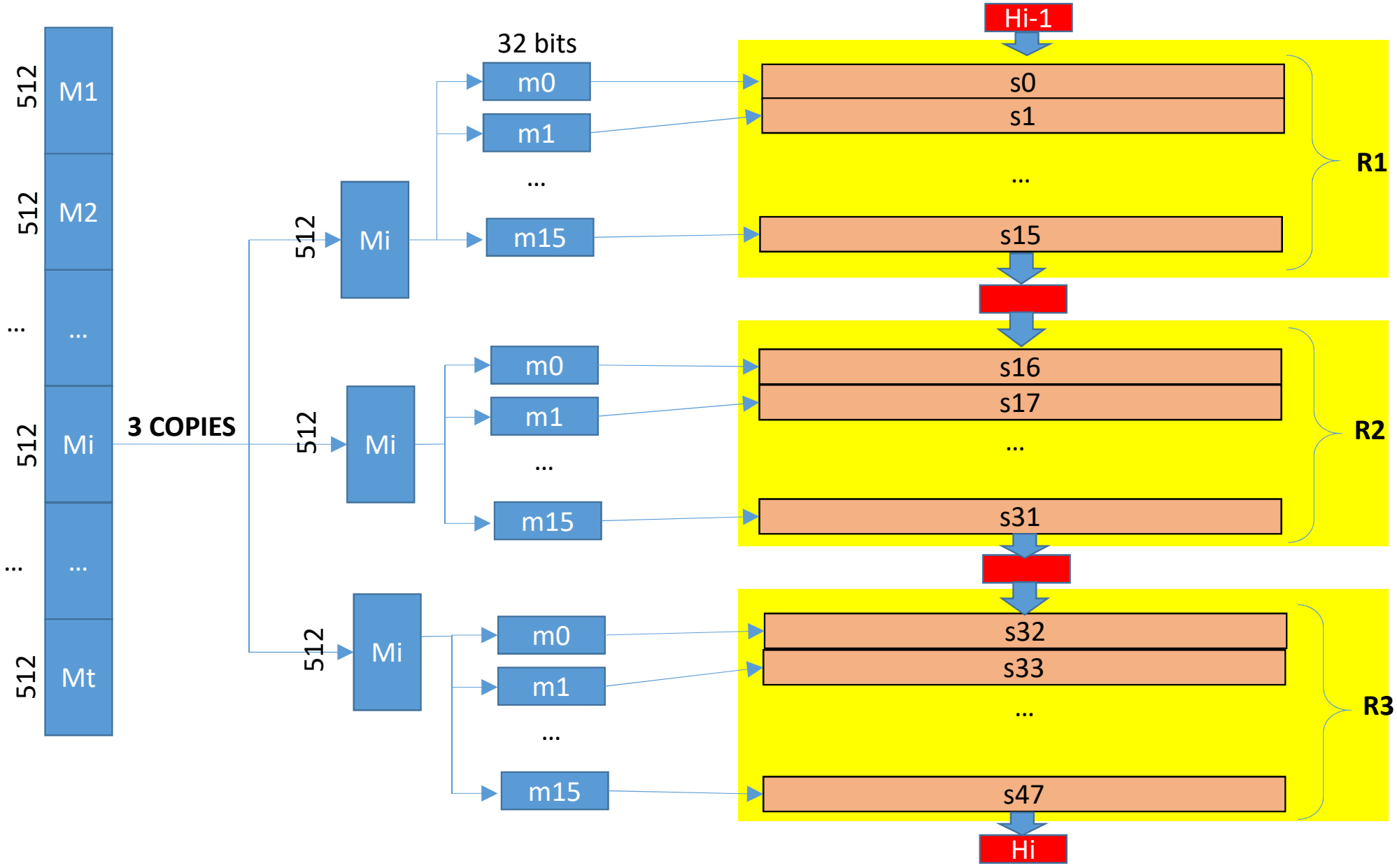


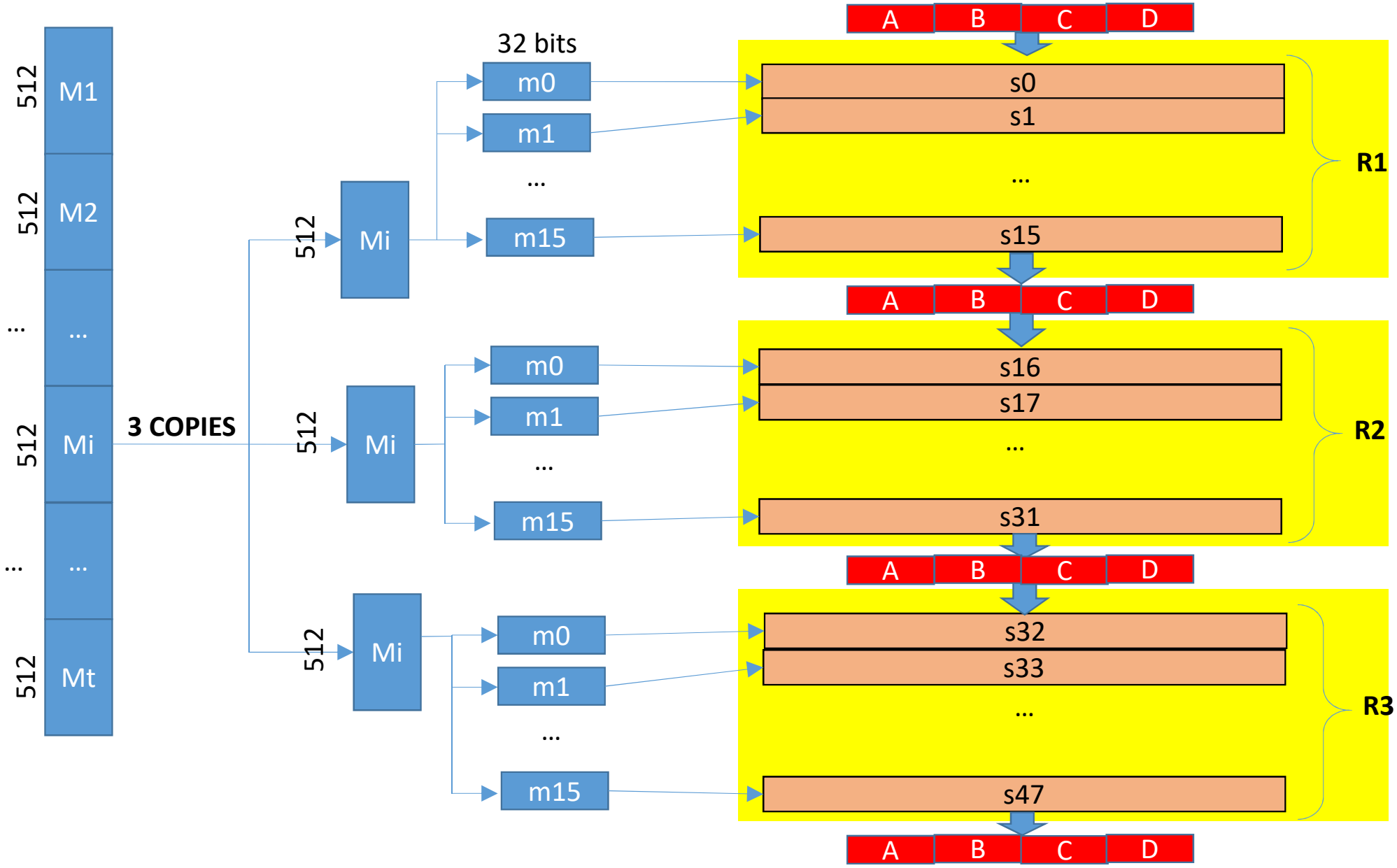
H0 (in Hex)	
A	67452301
B	efcdab89
C	98badcfe
D	10325476

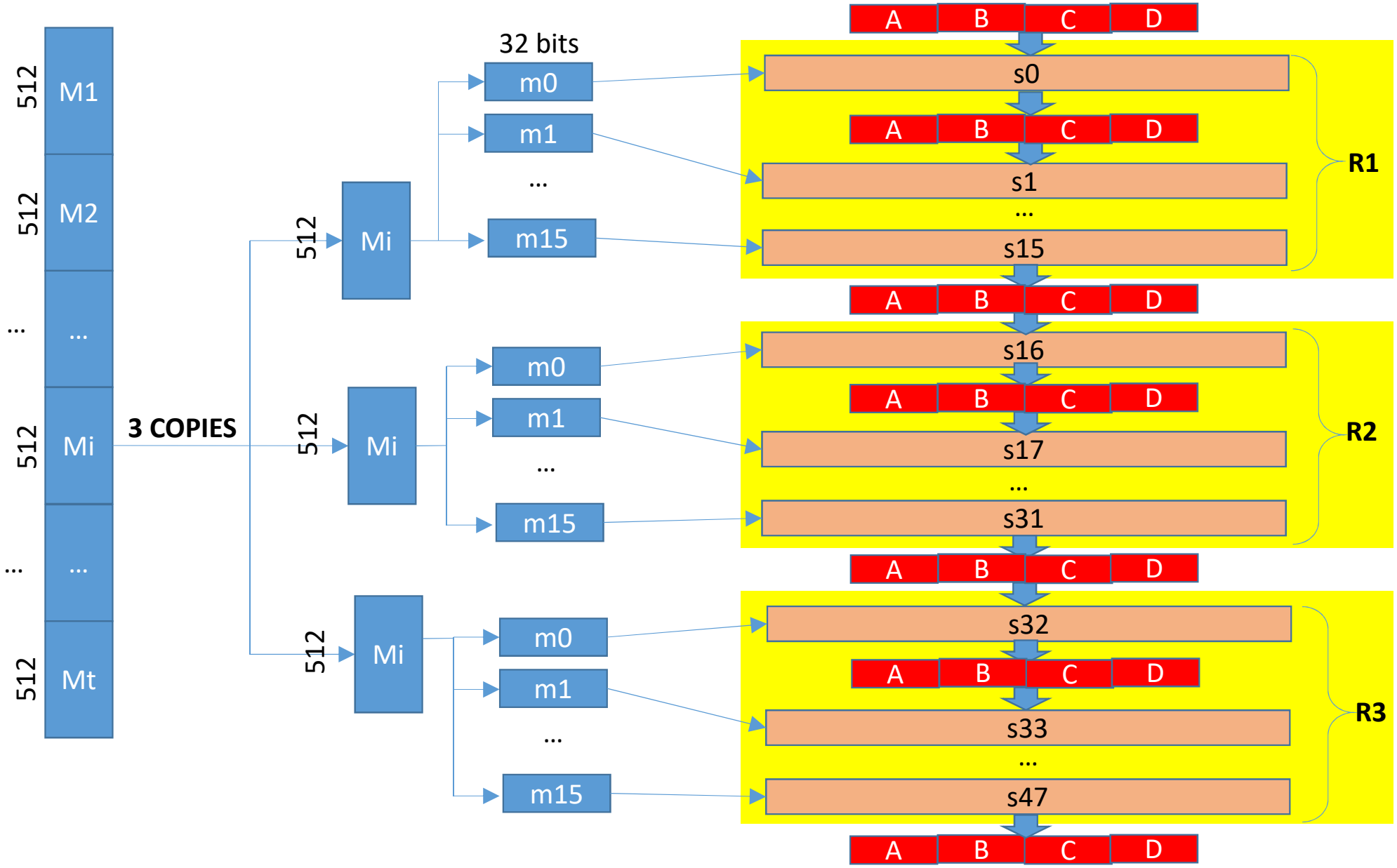

H0
R1
R2
R3
H1
...
Hi-1
R1
R2
R3
Hi
...
Ht-1
R1
R2
R3
Ht
Message
Digest
512
M1
512
M2
...
...
512
Mi
...
...
512
Mt

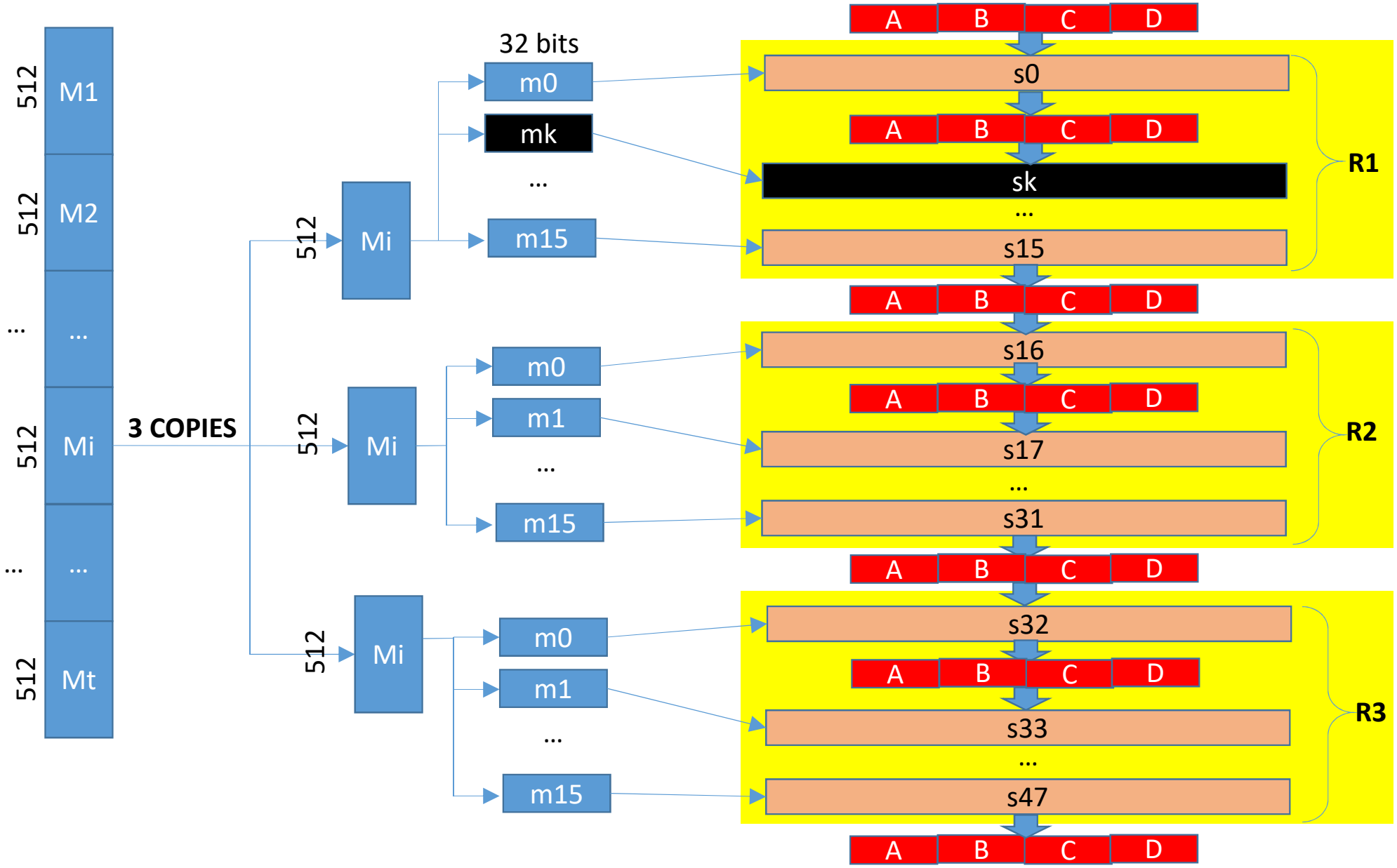

512
M1
512
M2
...
512
Mi
...
512
Mt
3 COPIES
512
Mi
512
Mi
512
Mi
...
H0
R1
R2
R3
H1
...
Hi-1
R1
R2
R3
Hi
...
Ht-1
R1
R2
R3
Ht
Message
Digest

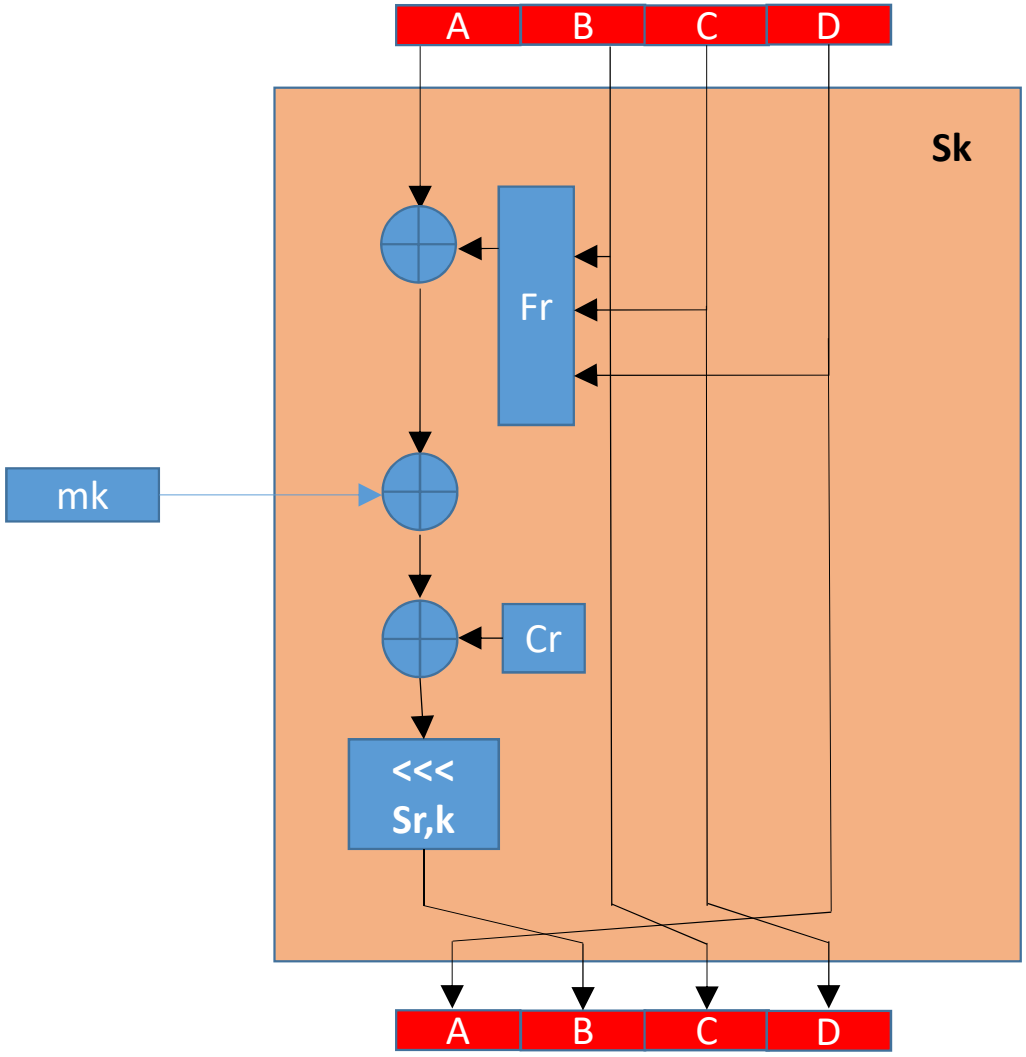

512
M1
512
M2
...
512
Mi
...
512
Mt
3 COPIES
512
Mi
512
Mi
512
Mi
32 bits
m0
m1
...
m15
m0
m1
...
m15
m0
m1
...
m15
Hi-1
s0
s1
...
s15
R1
s16
s17
...
s31
R2
s32
s33
...
s47
R3
Hi


512
M1
512
M2
...
512
Mi
...
512
Mt
3 COPIES
512
Mi
512
Mi
512
Mi
32 bits
m0
m1
...
m15
m0
m1
...
m15
m0
m1
...
m15
Hi-1
s0
s1
...
s15
R1
s16
s17
...
s31
R2
s32
s33
...
s47
R3
Hi


512
M1
512
M2
...
512
Mi
...
512
Mt
3 COPIES
512
Mi
512
Mi
512
Mi
32 bits
m0
m1
...
m15
m0
m1
...
m15
m0
m1
...
m15
A
B
C
D
s0
s1
...
s15
R1
A
B
C
D
s16
s17
...
s31
R2
A
B
C
D
s32
s33
...
s47
R3
A
B
C
D


512
M1
512
M2
...
512
Mi
...
512
Mt
3 COPIES
512
Mi
512
Mi
512
Mi
32 bits
m0
m1
...
m15
m0
m1
...
m15
m0
m1
...
m15
A
B
C
D
s0
s1
...
s15
R1
s16
s17
...
s31
R2
s32
s33
...
s47
R3

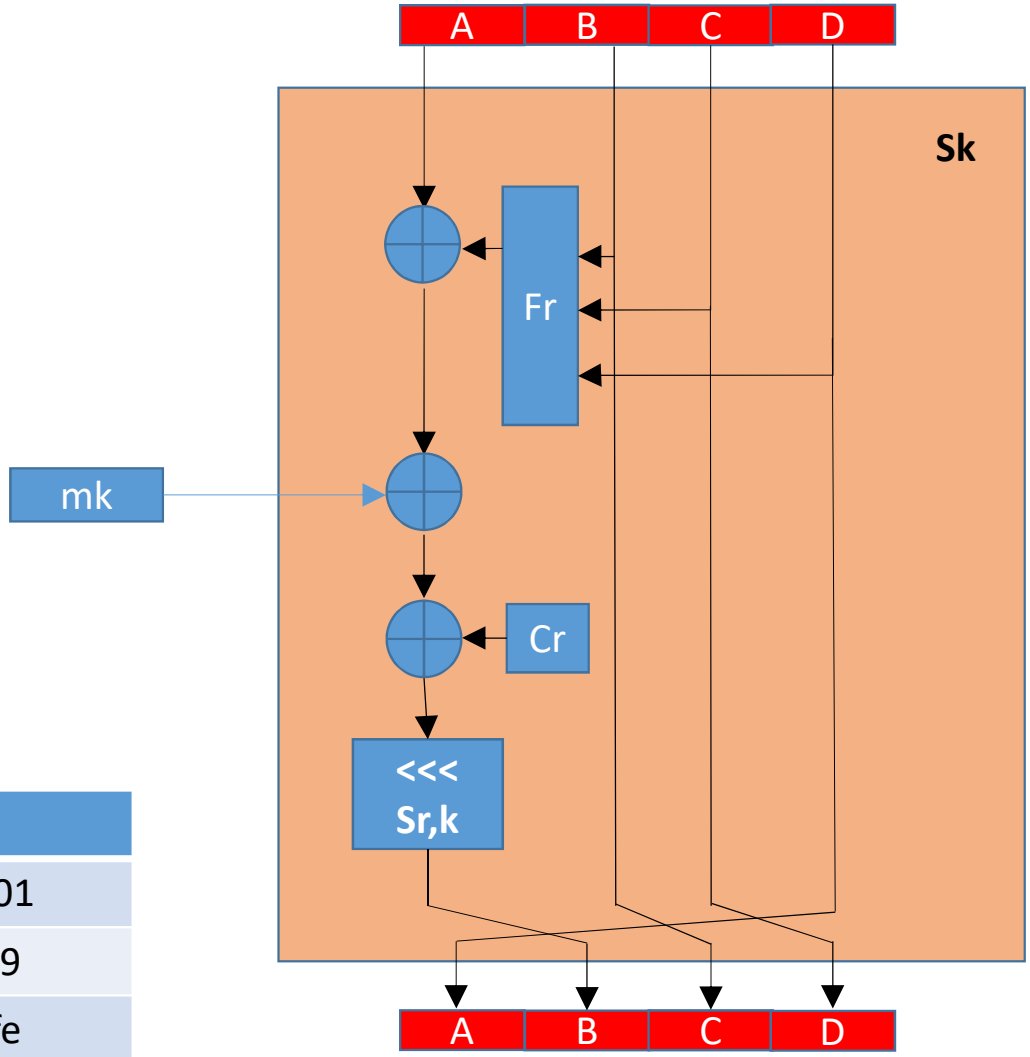

512
M1
512
M2
...
512
Mi
...
512
Mt
3 COPIES
512
Mi
512
Mi
512
Mi
32 bits
m0
mk
...
m15
m0
m1
...
m15
m0
m1
...
m15
A
B
C
D
s0
sk
...
s15
R1
s16
s17
...
s31
R2
s32
s33
...
s47
R3

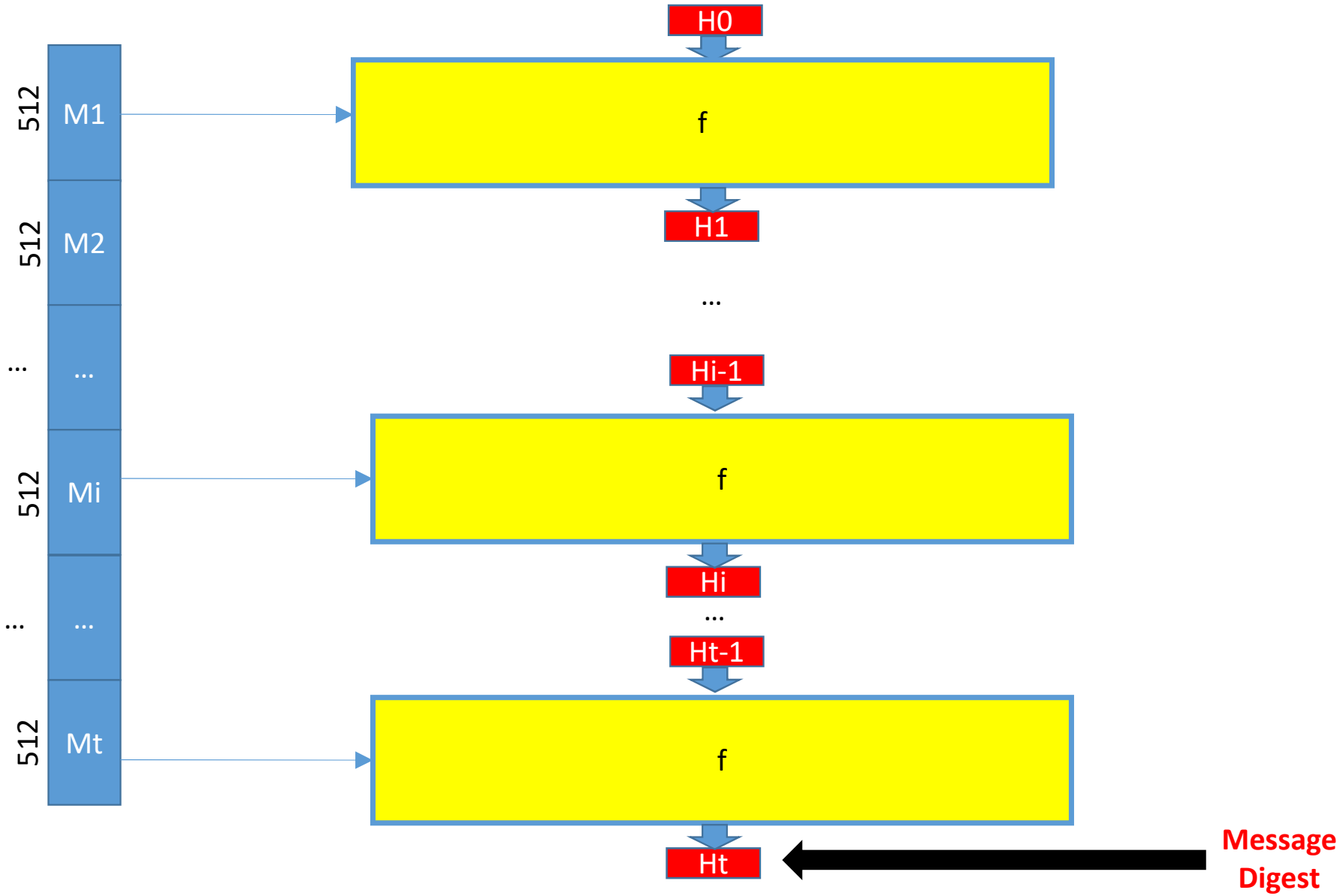

A
B
C
D
Sk
Fr
mk
Cr
<<<
Sr,k
A
B
C
D

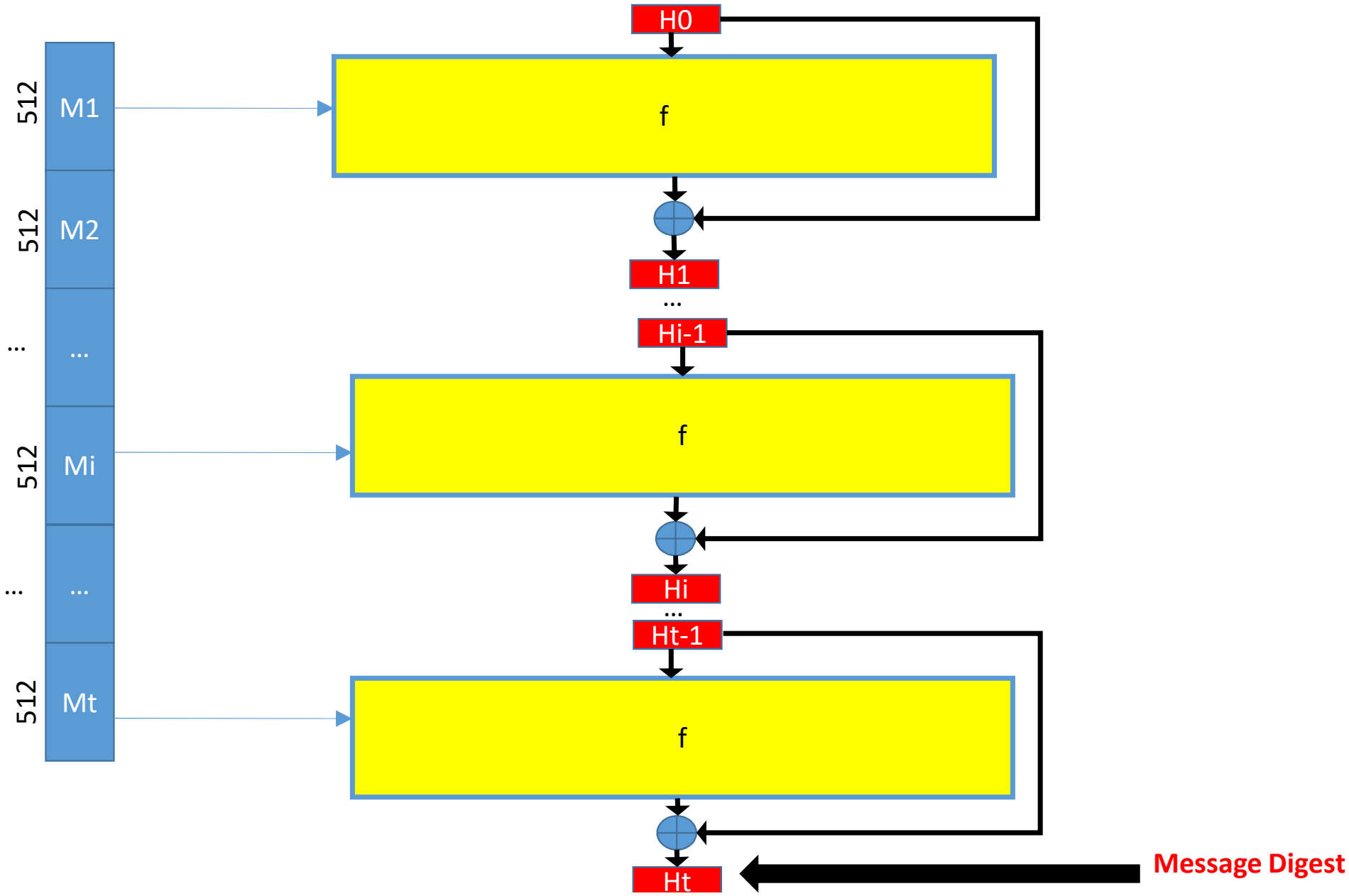
Round (r)	Fr	Cr (in Hex)	Sr,k
r = 1	IF	00000000	3,7,11,19, 3,7,11,19, 3,7,11,19, 3,7,11,19
r = 2	MAJ	5a827999	3,5,9,13, 3,5,9,13, 3,5,9,13, 3,5,9,13
r = 3	XOR	6ed9eba1	3,9,11,15, 3,9,11,15, 3,9,11,15, 3,9,11,15

Fr	Definition
IF(x,y,z)	$xy \oplus yz \oplus z$
MAJ(x,y,z)	$xy \oplus xz \oplus yz$
XOR(x,y,z)	$x \oplus y \oplus z$

H0 (Hex)	
A	67452301
B	efcdab89
C	98badcfe
D	10325476




H0
512
M1
f
H1
512
M2
...
...
Hi-1
512
Mi
f
Hi
...
...
Ht-1
512
Mt
f
Ht
Message
Digest

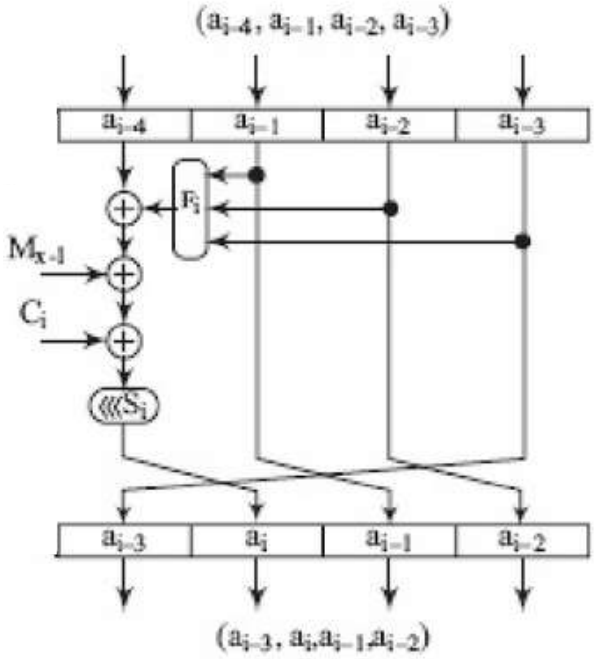

H0
512
M1
f
512
M2
H1
...
Hi-1
...
512
Mi
f
Hi
...
Ht-1
512
Mt
f
Ht
Message Digest

■ **Input Messages** One 512-bit input message block is denoted by $M=(m_0, m_1,\ldots, m_{15})$. Each message word m_k consists of 32 bits, which are denoted by m_{kj}, where $0 \le j \le 31$.

■ **Register Words** The 32-bit register words (or state variables) are denoted by a_i where i is the number of the compression step with $0 \le i \le 47$ and the register bits are indexed with a_{ij}, $0 \le j \le 31$. Each register word a_i is computed according to an update rule. The four output registers after each step i are grouped to $(a_{i-3}, a_i, a_{i-1}, a_{i-2})$.

In every step of MD4, a 32-bit register value (or often called state variable) a_i is computed according to the following recursive update rule:

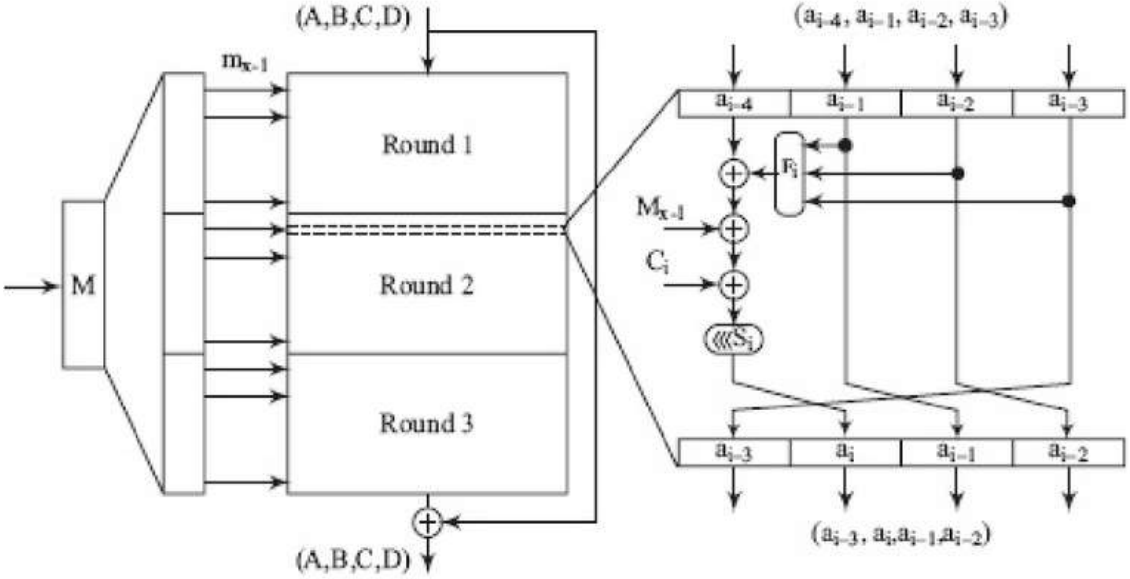
$$a_i = (a_{i-4} + f_i(a_{i-1},a_{i-1},a_{i-3}) + m_k + c_i) \ll s_i \quad 0 \le i \le 47$$



Each 512-bit message block of the padded message is compressed by the compression function which consists of three rounds having 16 steps each. In each round a different Boolean function f_i is used. The IF function is used for the first, the MAJ function for the second and the XOR function for the third round, as shown in the following table. Each message word m_k with $0 \le k \le 15$ of a message block M is added exactly once in each round.

$$XOR(x,y,z) = x \oplus y \oplus z$$
$$MAJ(x, y, z) = xy \oplus xz \oplus yz$$
$$IF(x,y,z) = xy \oplus xz \oplus z$$

(Step) i		f_i	c_i	circular left shift (rotation) by s_i positions s_i
Round 1	0...15	IF (x, y, z)	0x00000000	3, 7, 11, 19, 3, 7, 11, 19, 3, 7, 11, 19, 3, 7, 11, 19
Round 2	16...31	MAJ(x, y, z)	0x5a827999	3, 5, 9, 13, 3, 5, 9, 13, 3, 5, 9, 13, 3, 5, 9, 13
Round 3	21 . . . 47	XOR(x, y, z)	0x6ed9eba1	3, 9, 11, 15, 3, 9, 11, 15, 3, 9, 11, 15, 3, 9, 11, 15



For each message block, the update rule of the compression function is initialized by the chaining variables $(A,B,C,D) = (a_{-1}, a_{-4}, a_{-3}, a_{-2})$. The initial values for MD4 are

$$(A, B, C, D) = (0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476)$$

After the processing of all 48 steps, the last four register values are added to the chaining variables: $(A, B, C, D) = (A, B, C, D) + (a_{47}, a_{44}, a_{45}, a_{46})$. If no message block is processed anymore, the resulting hash value is the concatenation of the four chaining variables (A, B, C, D).