Practical key recovery attack against APOP, an MD5 based challenge response authentication. By Gaetan Leurent

Presented by:-
Raagi Sukhlecha
Lalit Agarwal

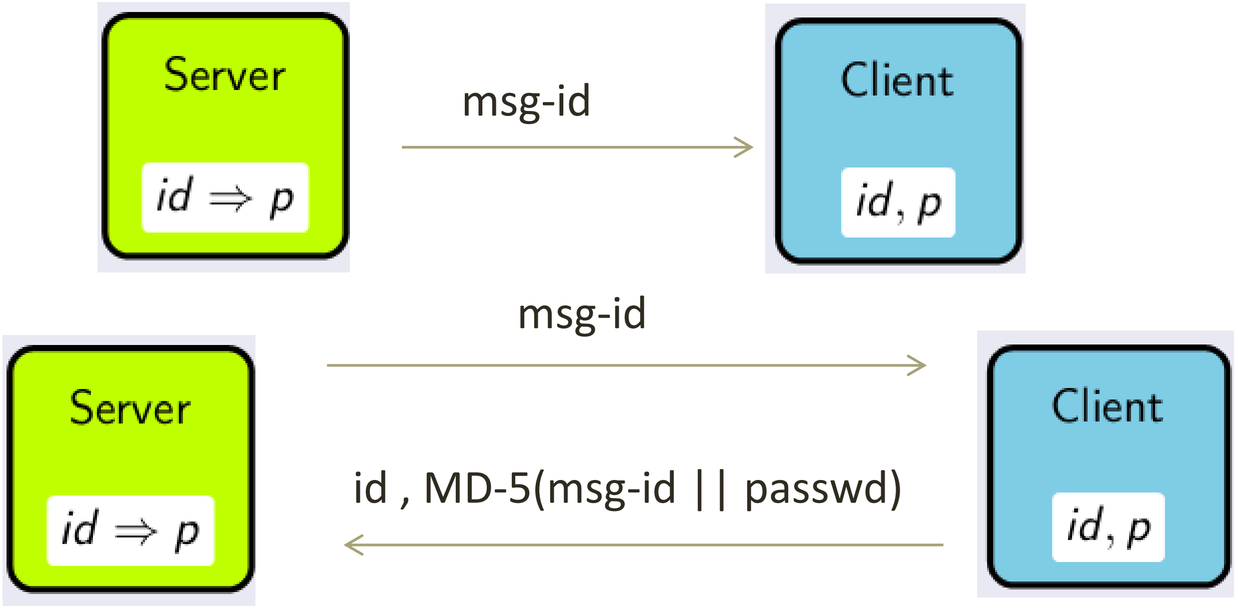
Guided By:-
Prof. Anish Mathuria

Outline

- Introduction
  - APOP – What is APOP and how does it work ?
  - MD-5 hashing algorithm
- APOP Attack
  - Abstract
  - Wang's attack on MD-5
  - Algorithm by Gaëtan Leurent
  - APOP Attack complexity
- APOP in practice

What is APOP ?

- Improvement to POP 3 which supported plain – text password
- APOP Provides simple challenges response authentication and avoids passive eavesdropping attack .
- It only does client authentication. No server authentication.



Example

According to RFC 1939 ,

1. The challenge should be enclosed with in <> with exactly one@ in between.
2. The remaining characters should be ASCII.
3. Inside the message-id, all characters are accepted, except:-
   1. 0x00 Null
   2. Ox3e Greater than Sign ('>')
   3. Ox0a Line-Feed
   4. Ox0d Carriage Return

MD-5 – Working

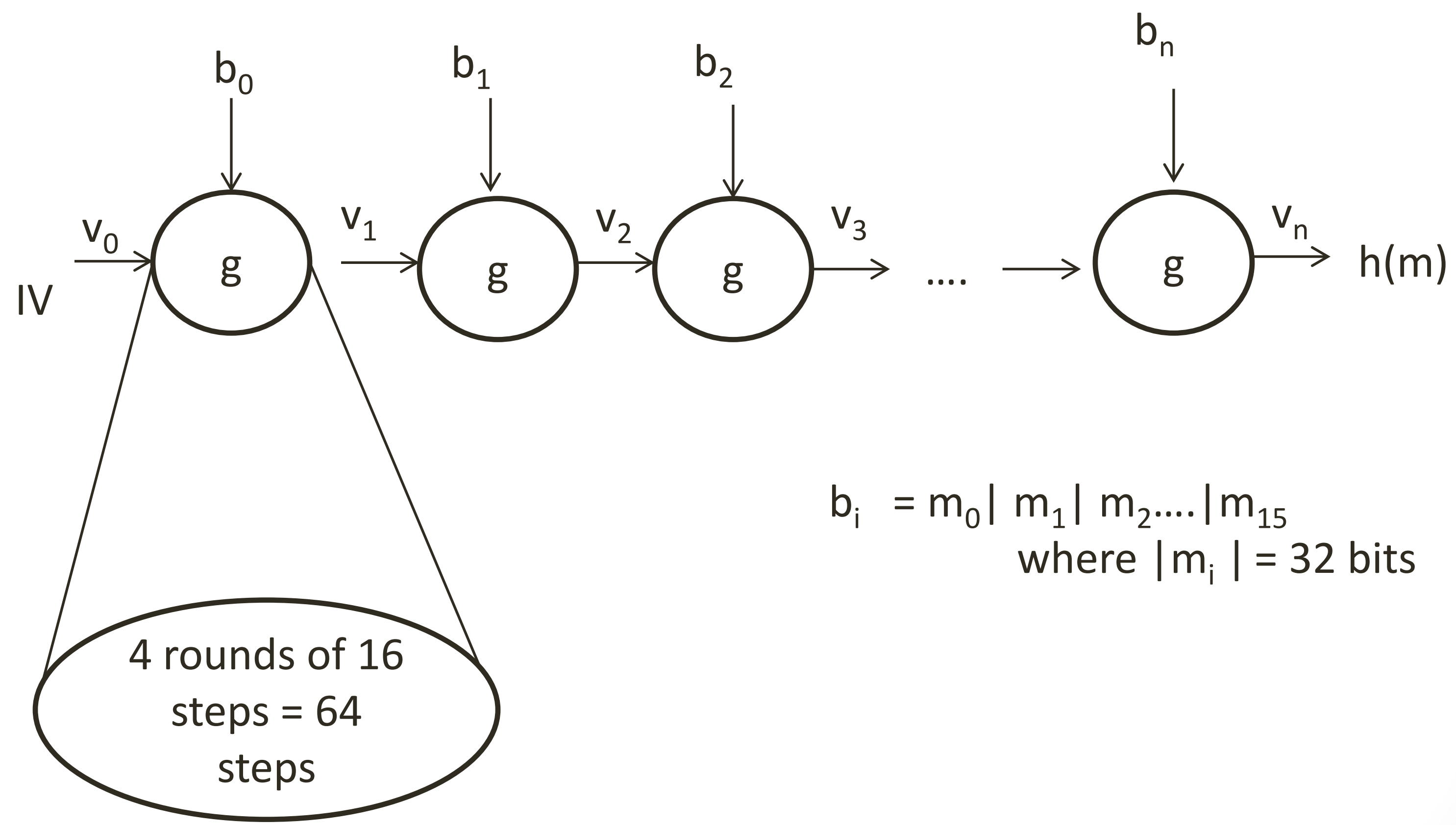
- Hashing algorithm; uses Merkle damgard construction
- Message blocks of 512 bits and initialization vector IV of 128 bits.
- Uses bitwise functions
  - additions mod 2^{32} : +
  - Boolean functions: f_i
  - Rotations : $<< s_i$

Consider a message M

$M \xrightarrow{\text{padding}} b_0 | b_1 | b_2 .... | b_n$ where $|b_i| = 512$ bits

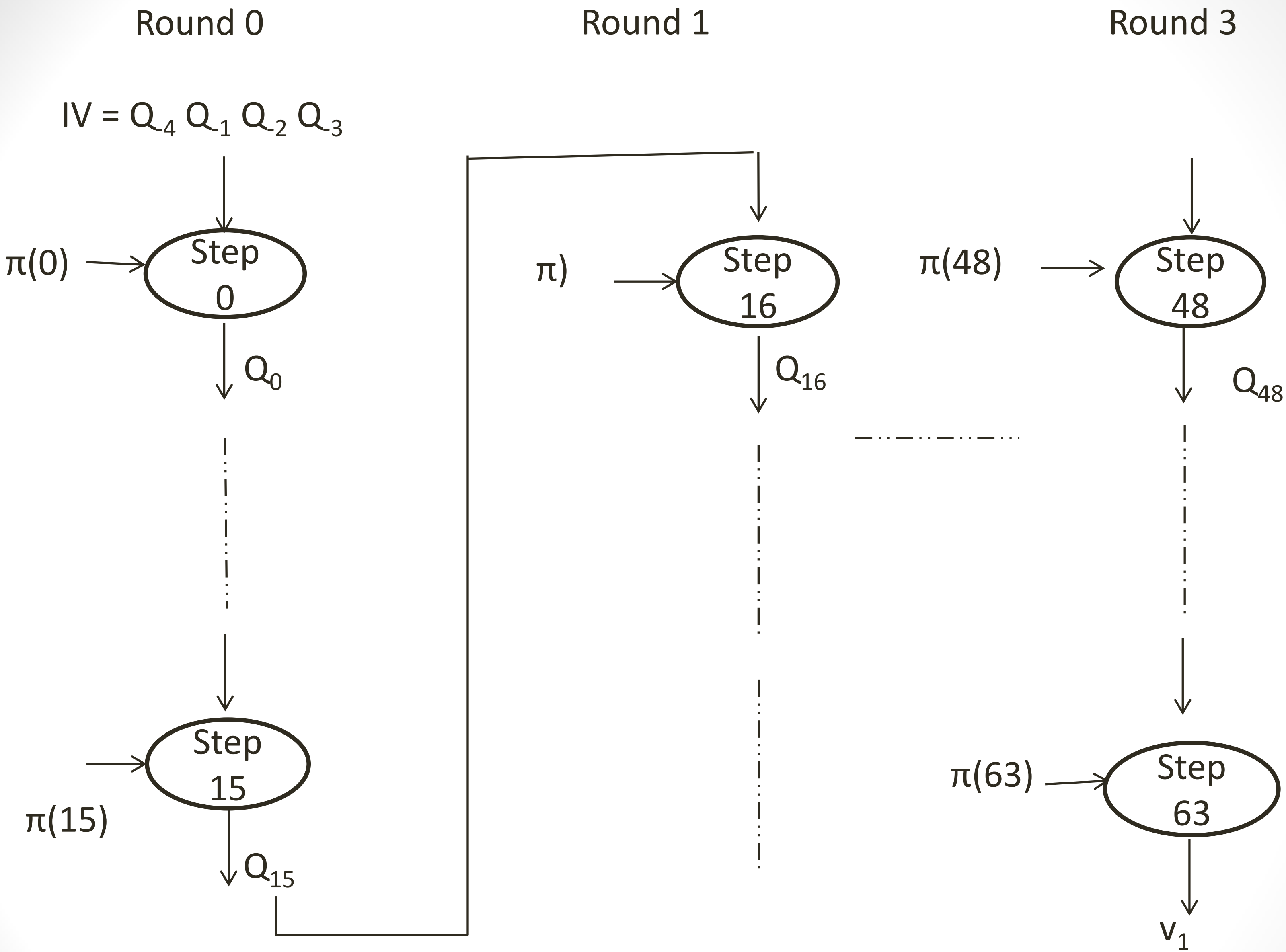
g - compression function

MD-5 – Working (cont.)



$b_i = m_0 | m_1 | m_2 …. | m_{15}$

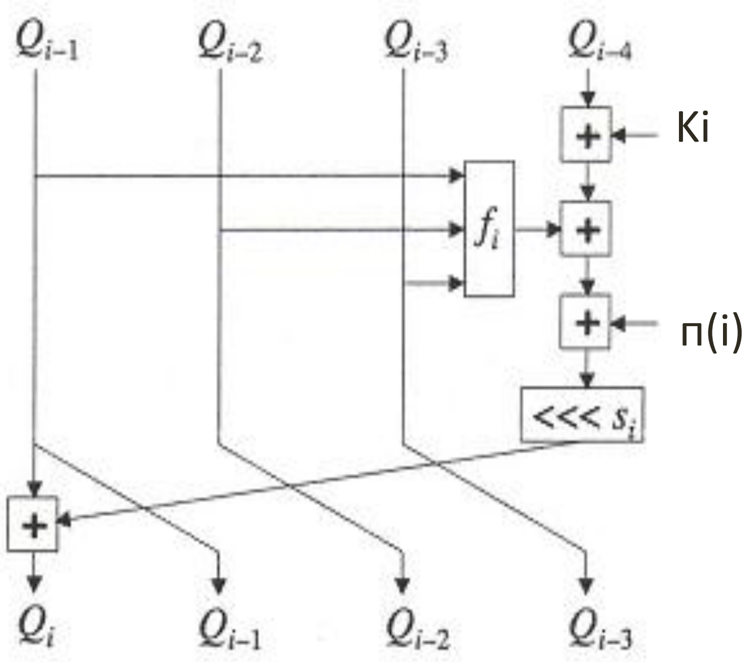
where $|m_i| = 32$ bits



Where IV is broken into 4 32 bit words Q_{-4} Q_{-1} Q_{-2} Q_{-3}

Q_i is the output of each step i (0<= i <=63)

MD-5 – Working (cont.) - A MD-5 step



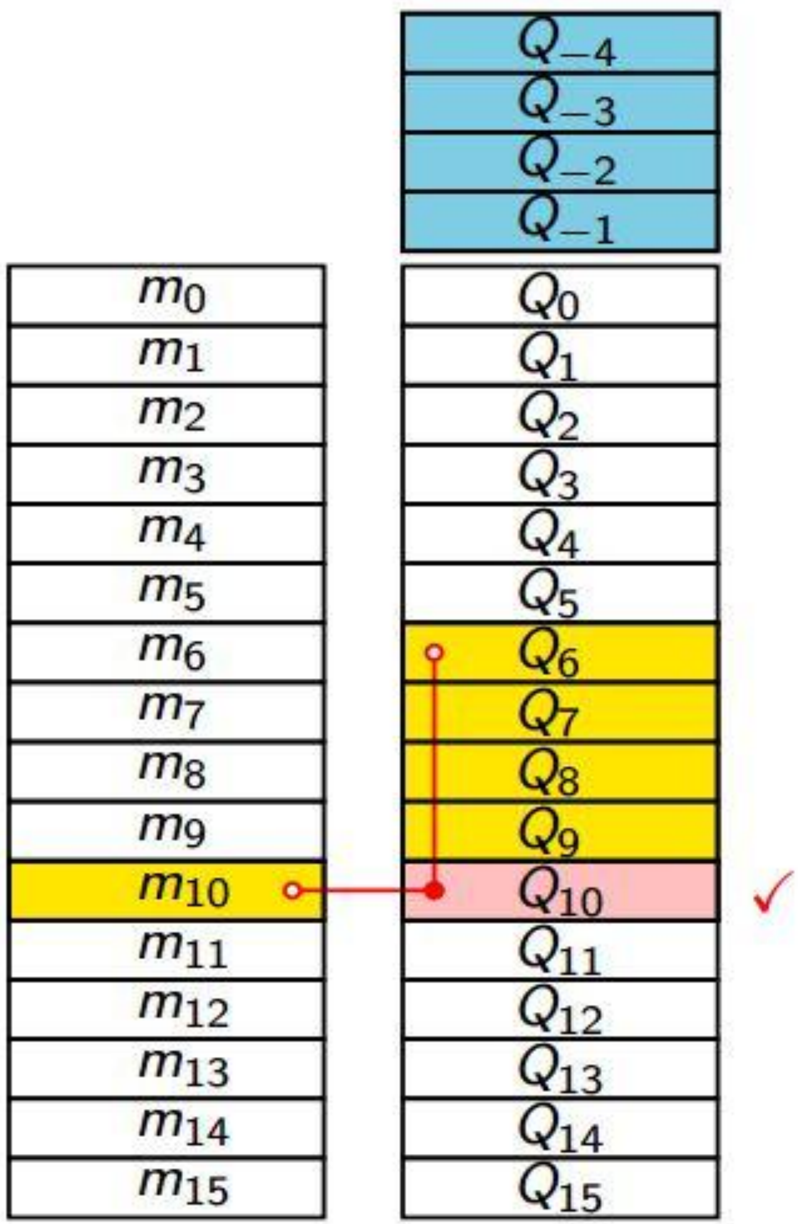
where

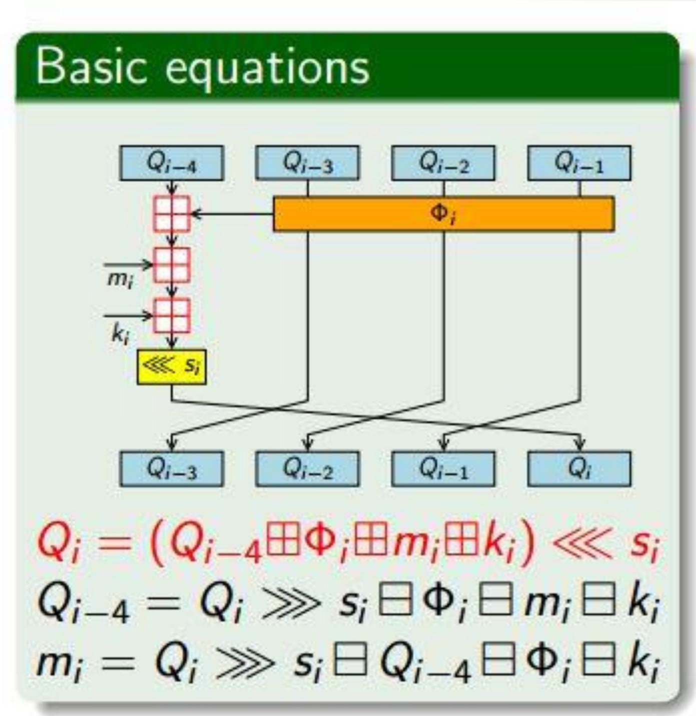
- s_i and K_i as predefined constant
- π(i) is permutation applied to input blocks
- f_i as functions defined as

$$f_i(A,B,C) = \begin{cases} F(A,B,C) & \text{if } 0 \le i \le 15 \\ G(A,B,C) & \text{if } 16 \le i \le 31 \\ H(A,B,C) & \text{if } 32 \le i \le 47 \\ I(A,B,C) & \text{if } 48 \le i \le 63. \end{cases}$$

$$F(A,B,C) = (A \wedge B) \vee (\neg A \wedge C)$$
$$G(A,B,C) = (A \wedge C) \vee (B \wedge \neg C)$$
$$H(A,B,C) = A \oplus B \oplus C$$
$$I(A,B,C) = B \oplus (A \vee \neg C)$$

Basic Equation

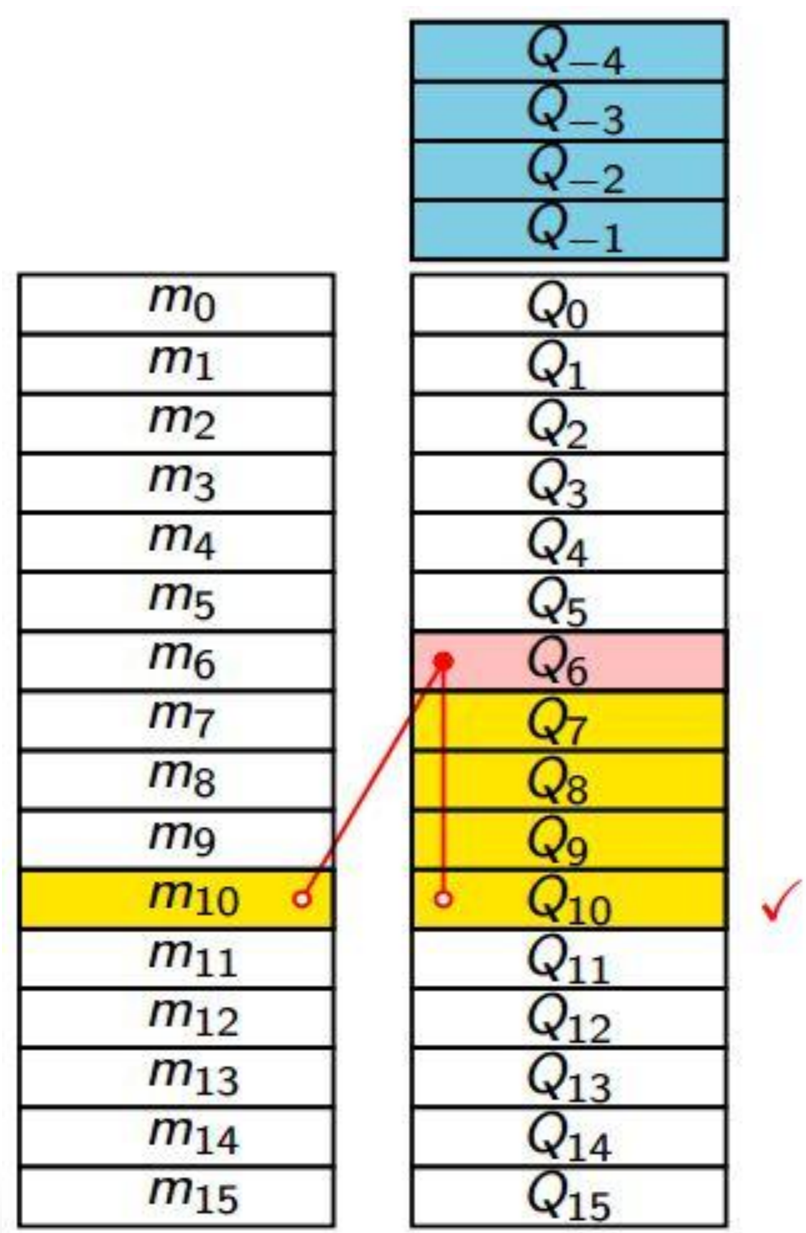


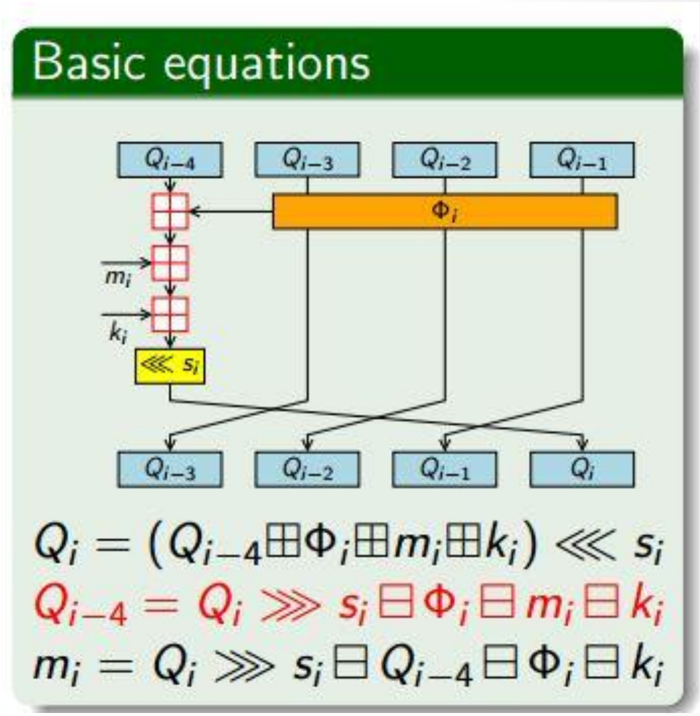


If Q_i, Q_{i+1}, Q_{i+2}, Q_{i+3} are known, then we can compute Q_{i+4}.

Here we compute $\mathbf{Q_{10}}$ from Q_6, Q_7, Q_8, Q_9 and m_{10}.

Basic Equation

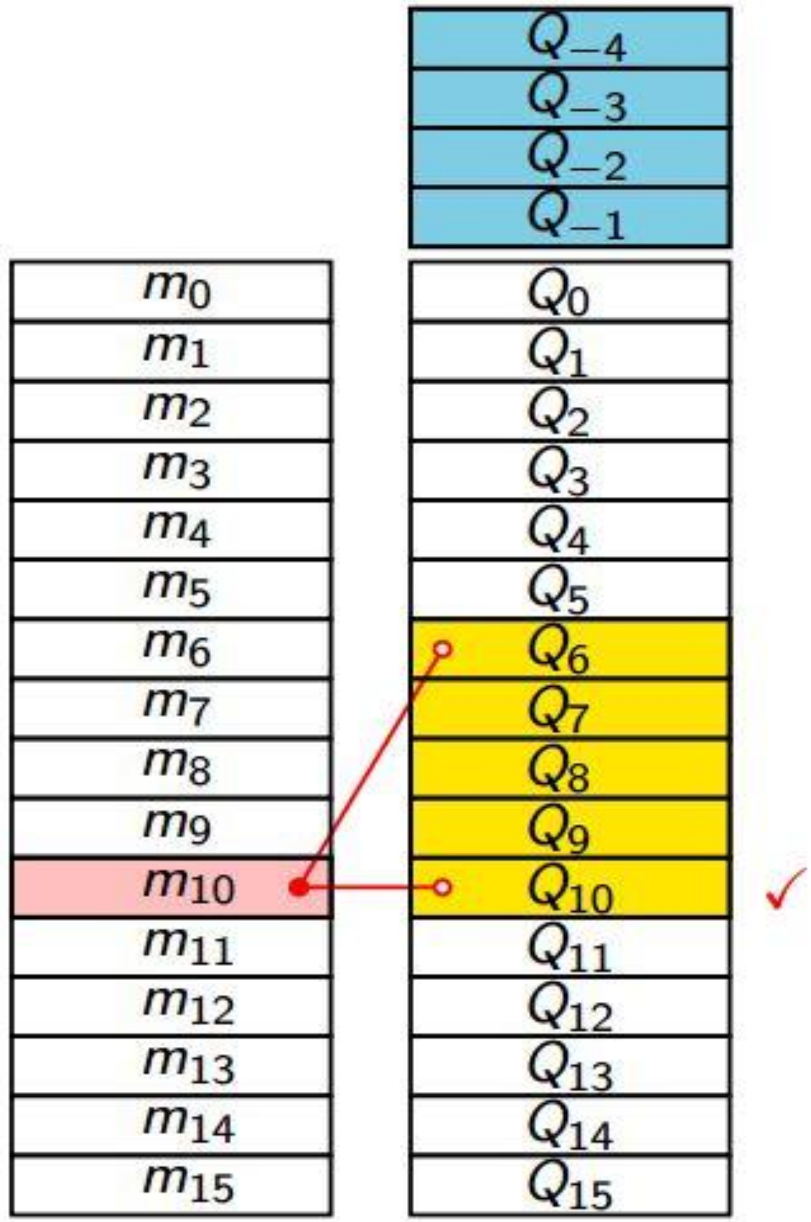




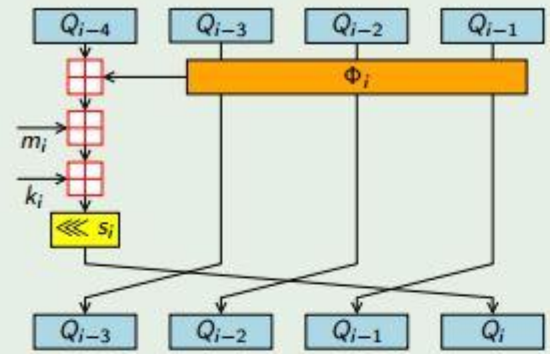
If Q_{i+1}, Q_{i+2}, Q_{i+3}, Q_{i+4} are known, then we can compute Q_i.

Here we compute $\mathbf{Q_6}$ from Q_7, Q_8, Q_9, Q_{10} and m_{10}.

Basic Equation



Basic equations



$$Q_i = (Q_{i-4} \boxplus \Phi_i \boxplus m_i \boxplus k_i) \lll s_i$$

$$Q_{i-4} = Q_i \ggg s_i \boxminus \Phi_i \boxminus m_i \boxminus k_i$$

$$m_i = Q_i \ggg s_i \boxminus Q_{i-4} \boxminus \Phi_i \boxminus k_i$$

If Q_i - Q_{i-4} are known then we can compute m_i.

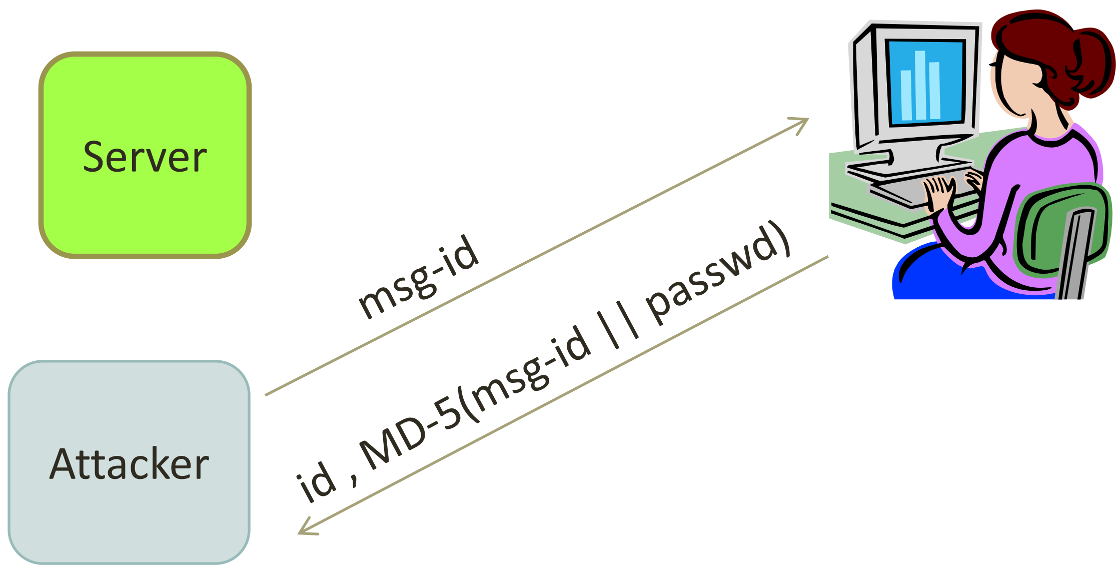
Here we compute $\mathbf{m_{10}}$ from Q_6 and Q_{10}.

APOP Attack

- Abstract
- Wang’s Attack
  - Wang’s attack on MD-4 and MD-5
  - Problem with Wang’s attack
- Algorithm by Gaëtan Leurent
- Message freedom
- APOP Attack Complexity

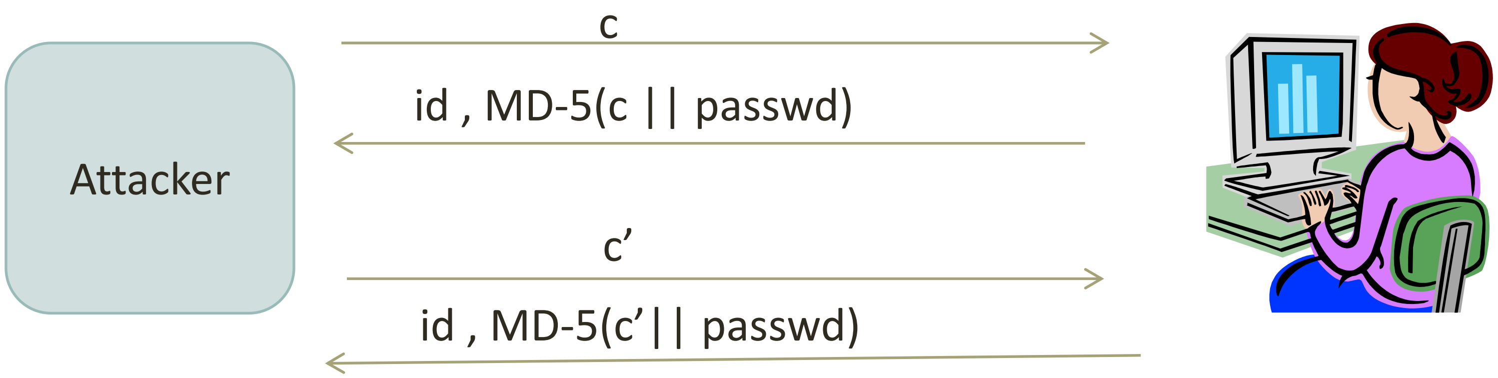
Abstract of the attack

- **Goal:-** To recover some characters of the client's password
- Attacker impersonates server and sends crafted challenge



Abstract of the attack (cont.)

- Attacker sends challenges in such a way that hashed responses will collide if the part of the password was rightly guessed



Attack

	Block 1	Block 2		Challenge
M =	<?????????...........@	...........????????????>	x	C = <?????...??>
M' =	<///////..............@	.........................//////////>	x	C' = </////...//>

H(M) = H(M')

R = MD-5 (	<?????????...........@	...........????????????>	p_0	p_1 p_2 p_3.......................pad	)
R' = MD-5 (	<///////..............@	.........................//////////>	p_0	p_1 p_2 p_3.......................pad	)

R and R' are equal if p_0 = x

To test the first password character, the attacker will construct pairs to test each of the 256 ASCII values .

Note:- The collision is unlikely if p_0 != x ?

Attack (cont.)

	Block 1	Block 2		Challenge
M =	<?????????...........@	...........???????????>p0	y	C = <???....?>
M' =	<///////..............@	........................//////>p0	y	C' = <///...//>

H(M) = H(M')

R = MD-5 (	<?????????...........@	...........????????????>p0	p_1	p_2 p_3......................pad	)
R' = MD-5 (	<///////..............@	........................//////>p0	p_1	p_2 p_3......................pad	)

Both hashes collide if p_1= y
To test the second password character, pairs to test 256 ASCII values have to be constructed

Questions ?????

- How can we fix the last message word ?
- Does that mean that we can recover the entire message ? If not how many characters can we recover .
- What will be the time complexity of it ?
- Can APOP be still used ?
- APOP being an offline protocol , is this attack meaningful ?

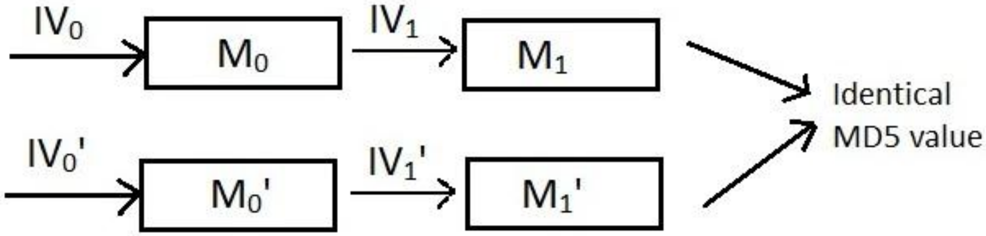
Wang's Attack

- In 2004, Xiaoyun Wang published a MD5 collision. Did not reveal anything about the attack.

- Determined two 1024-bit messages

  $M' = (M'_0, M'_1)$ and $M = (M_0, M_1)$

  where M_0', M_1', M_0, M_1 are each 512-bit blocks.
  So that MD5 hashes of the two messages are the same



- Reverse engineering – revealed many aspects of attack; improvements in attack

Wang's Attack

Modular Difference, Δy

Consider bytes

$y' = 00010101$ and $y = 00000101$

$z' = 00100101$ and $z = 00010101$

Note that

$y' - y = z' - z = 00010000 = 24$

Then wrt modular subtraction, these pairs are **indistinguishable.**

Signed difference, ∇y=y'-y

Denote $y'_i=1$, $y_i=0$ as "+"

Denote $y'_i=0$, $y_i=1$ as "–"

Denote $y'_i=y_i$ as "."

Consider bytes

$z' = 10100101$ and $z = 10010101$

Then ∇z is "..+-...."

It is more **restrictive** than modular subtraction.

Wang's Attack

- **Step 1: Specify Input Differential Pattern**

  - Applies to input M and M'.
  - Uses Modular Difference.

$$\Delta M_0 = M'_0 - M_0 = (0,0,0,0,2^{31},0,0,0,0,0,0,2^{15},0,0,2^{31},0)$$
$$\Delta M_1 = M'_1 - M_1 = (0,0,0,0,2^{31},0,0,0,0,0,0,-2^{15},0,0,2^{31},0)$$

- Note: M'_0 and M_0 differ only in words 4, 11 and 14
- Note: M'_1 and M_1 differ only in words 4, 11 and 14

  - Now, we only need to find M. Then M′ can be determined by the differential.

$$M'_0 = M_0 + \Delta M_0 \quad \text{and} \quad M'_1 = M_1 + \Delta M_1$$

Wang's Attack

Input vector 1:

d1	31	dd	02	c5	e6	ee	c4	69	3d	9a	06	98	af	f9	5c
2f	ca	b5	87	12	46	7e	ab	40	04	58	3e	b8	fb	7f	89
55	ad	34	06	09	f4	b3	02	83	e4	88	83	25	71	41	5a
08	51	25	e8	f7	cd	c9	9f	d9	1d	bd	f2	80	37	3c	5b
d8	82	3e	31	56	34	8f	5b	ae	6d	ac	d4	36	c9	19	c6
dd	53	e2	b4	87	da	03	fd	02	39	63	06	d2	48	cd	a0
e9	9f	33	42	0f	57	7e	e8	ce	54	b6	70	80	a8	0d	1e
c6	98	21	bc	b6	a8	83	93	96	f9	65	2b	6f	f7	2a	70

Input vector 2:

d1	31	dd	02	c5	e6	ee	c4	69	3d	9a	06	98	af	f9	5c
2f	ca	b5	07	12	46	7e	ab	40	04	58	3e	b8	fb	7f	89
55	ad	34	06	09	f4	b3	02	83	e4	88	83	25	f1	41	5a
08	51	25	e8	f7	cd	c9	9f	d9	1d	bd	72	80	37	3c	5b
d8	82	3e	31	56	34	8f	5b	ae	6d	ac	d4	36	c9	19	c6
dd	53	e2	34	87	da	03	fd	02	39	63	06	d2	48	cd	a0
e9	9f	33	42	0f	57	7e	e8	ce	54	b6	70	80	28	0d	1e
c6	98	21	bc	b6	a8	83	93	96	f9	65	ab	6f	f7	2a	70

Identical MD5 value: 79054025255fb1a26e4bc422aef54eb4

Wang's Attack

- **Step 2: Specify Output Differential Pattern**

  - Applies to intermediate values, Q'_i and Q_i
  - Uses signed difference. Hence very restrictive.
  - Most mysterious part of the attack.

j	Output	W_j	ΔW_j	ΔOutput	∇Output
4	Q_4	X_4	2^{31}	$\bar{6}$	....-++++++ ++++++++ ++......
5	Q_5	X_5	0	$\overset{+}{31}\,\overset{+}{23}\,\overset{-}{6}$	+....... +....... ....-......
6	Q_6	X_6	0	$\overset{-}{27}\,\overset{+}{23}\,\overset{-}{6}\,\overset{-}{0}$	++++++-- -....... ....-+++ ++-+++++
7	Q_7	X_7	0	$\overset{-}{23}\,\overset{-}{17}\,\overset{-}{15}\,\overset{+}{0}$	.... -..-+++- +....... ....+
8	Q_8	X_8	0	$\overset{+}{31}\,\overset{-}{6}\,\overset{+}{0}$	-....... ....- ++....+-

- j determines the step number
- Q_i are outputs for M_0
- ΔW_j are input (modular) differences
- ΔOutput is output modular difference
- ∇Output is output signed ("precise") difference

Wang’s Attack

- **Step 3: Derive a set of sufficient conditions**

	Conditions on M_0	Number
Q_2	....0... ....0... .0......	3
Q_3	1....... 0^^^1^^^ ^^^^1^^^ ^011....	21
Q_4	1000100. 01..0000 00000000 0010.1.1	27
Q_5	0000001^ 01111111 10111100 0100^0^1	32
Q_6	00000011 11111110 11111000 00100000	32
Q_7	00000001 1..10001 0.0.0101 01000000	28
Q_8	11111011 ...10000 0.1^1111 00111101	28
Q_9	0111.... 0..11111 1101...0 01....00	19
Q_{10}	00100000 1...0001 11000000 11000010	29
Q_{11}	000...00 ....1000 0001...1 0.......	15
Q_{12}	01....01 ....1111 111....0 0...1...	14
Q_{13}	0.0...00 ....1011 111....1 1...1...	14
Q_{14}	0.1...01 ....0 1....... ....0...	7
Q_{15}	0!1..... ......!.	4
Q_{16}	0!...... ......0. ^....... ....^...	5
Q_{17}	0.^..... ......1.	3
Q_{18}	0....... ......0.	2
Q_{19}	0....... .....!..	2
Q_{20}	0....... ......^.	2
	Subtotal	287

Wang's Attack

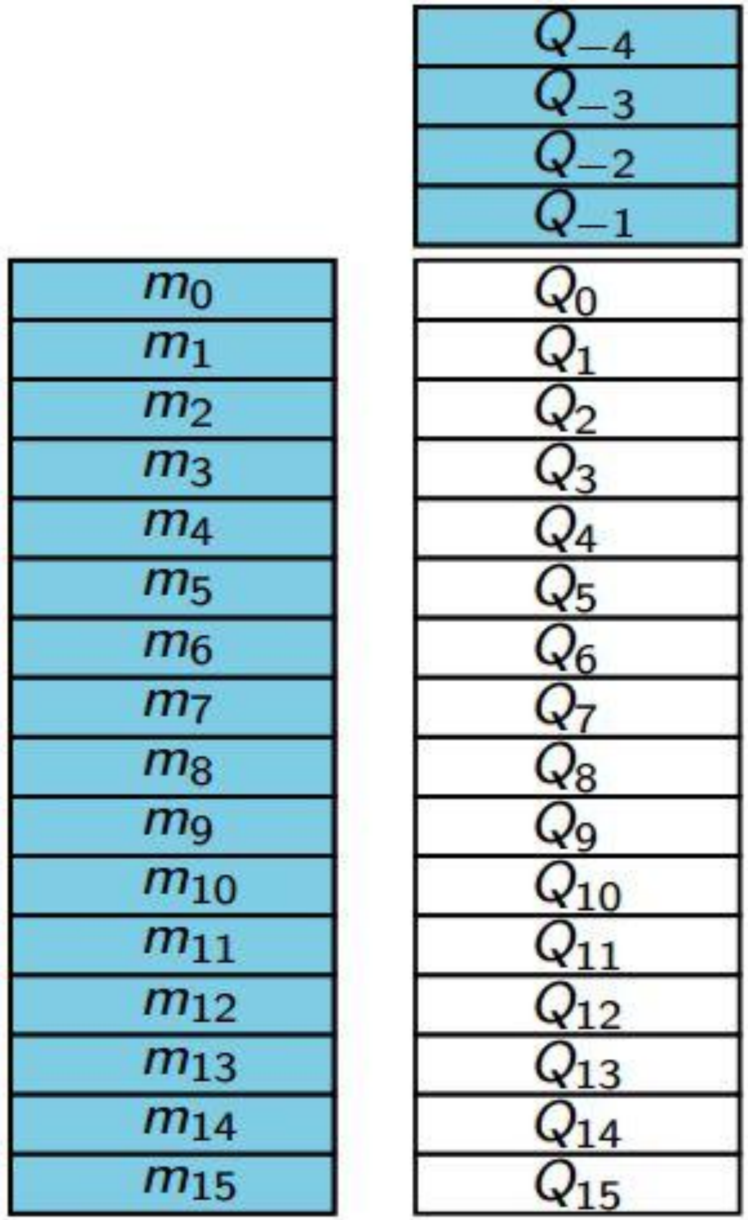
- **Step 4: Find a set of messages which satisfy all the conditions in step3.**

  - Generate random 512-bit M_0
  - Modify the message so that all the conditions hold.
  - Follow similar procedure to find M_1
  - Compute M'_0 and M'_1 using

$$M'_0 = M_0 + \Delta M_0 \quad \text{and} \quad M'_1 = M_1 + \Delta M_1$$

Now $H(M) = H(M')$

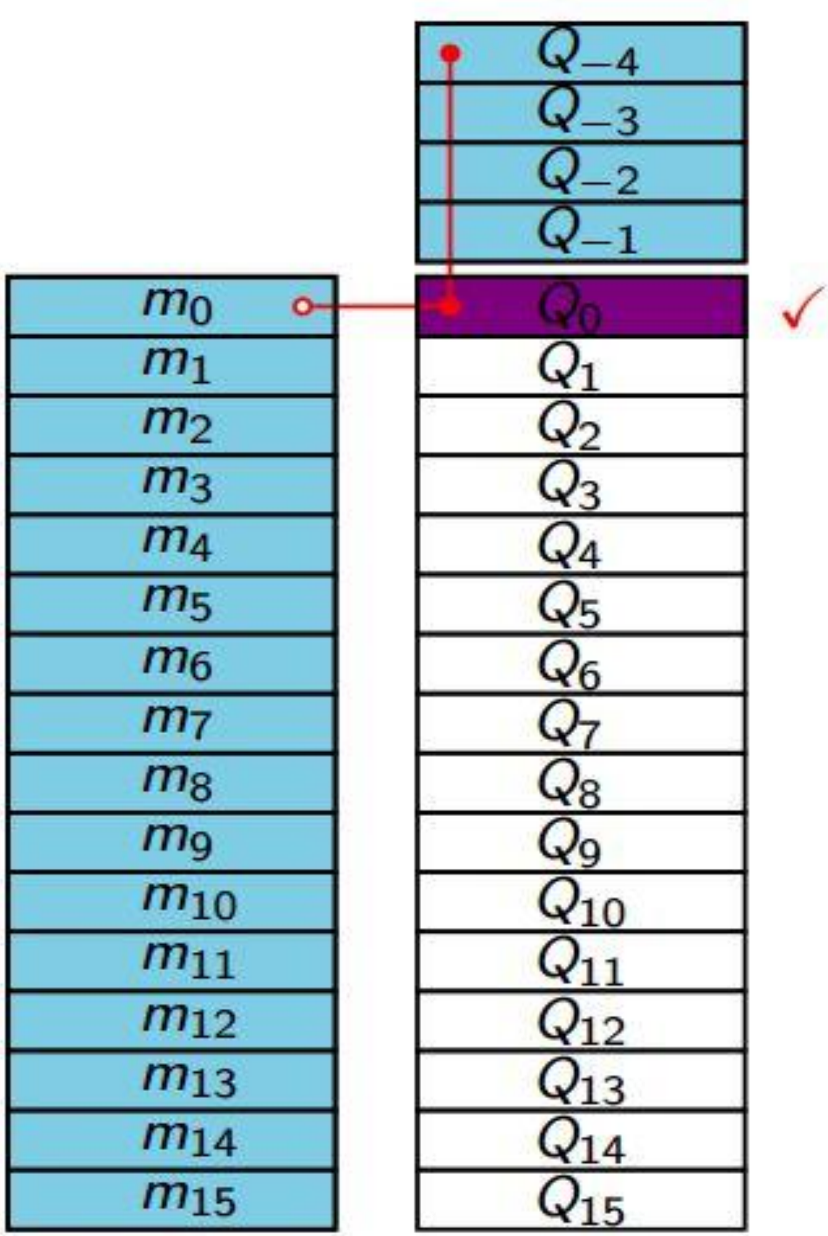
Wang's Approach to satisfy conditions in the first round



Message Modification

- ***Select a message m_i***
- Compute the corresponding Q_i
- Modify Q_i to satisfy the conditions. Recompute m_i

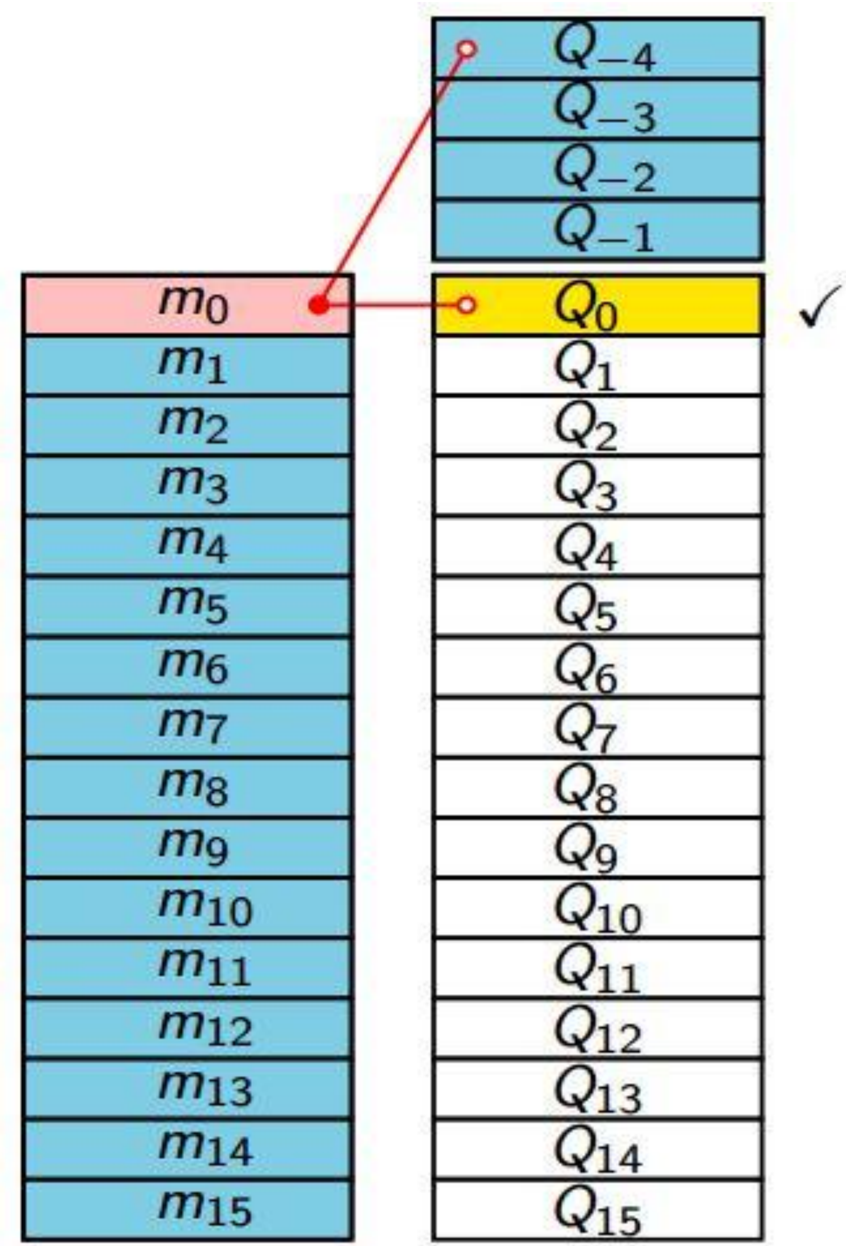
Wang's Approach to satisfy conditions in the first round



Message Modification

- *Select a message m_i*
- **Compute the corresponding Q_i**
- Modify Q_i to satisfy the conditions. Recompute m_i

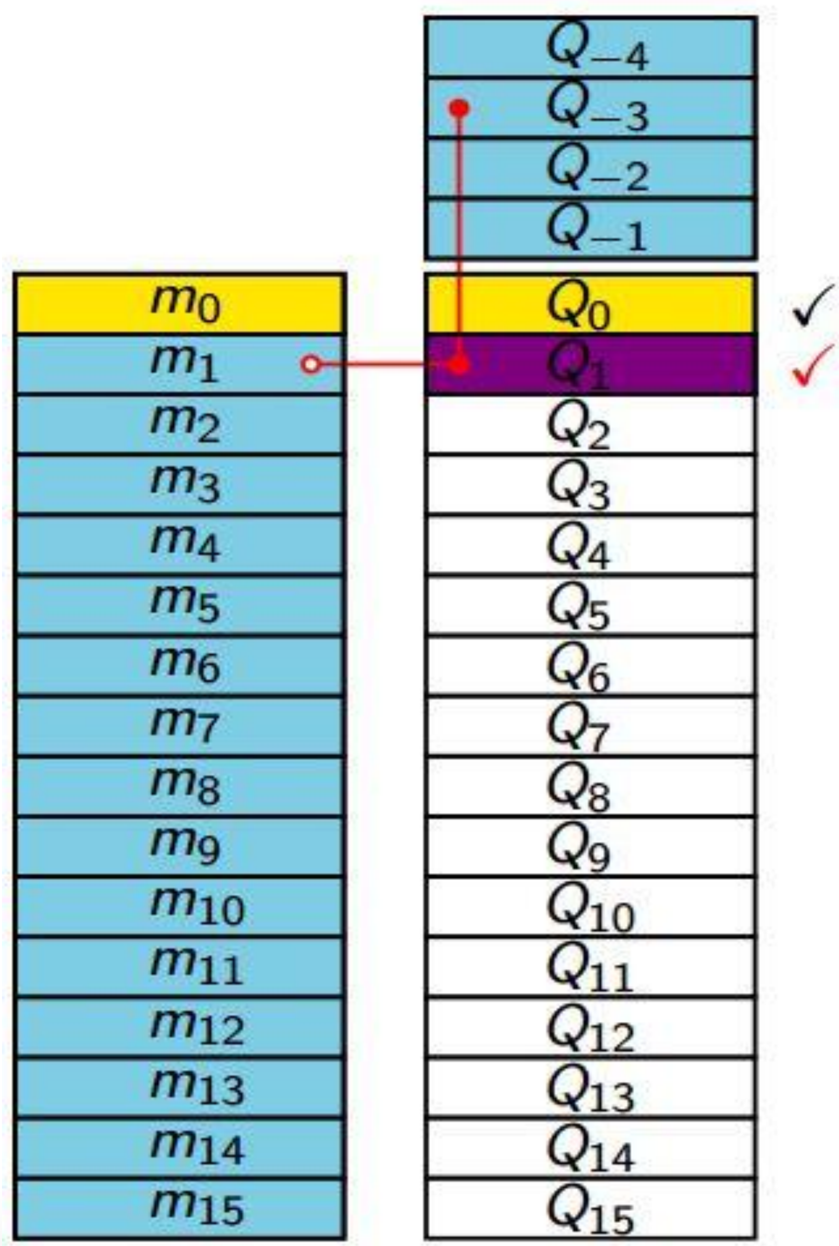
Wang's Approach to satisfy conditions in the first round



Message Modification

- *Select a message m_i*
- Compute the corresponding Q_i
- **Modify Q_i to satisfy the conditions. Recompute m_i**

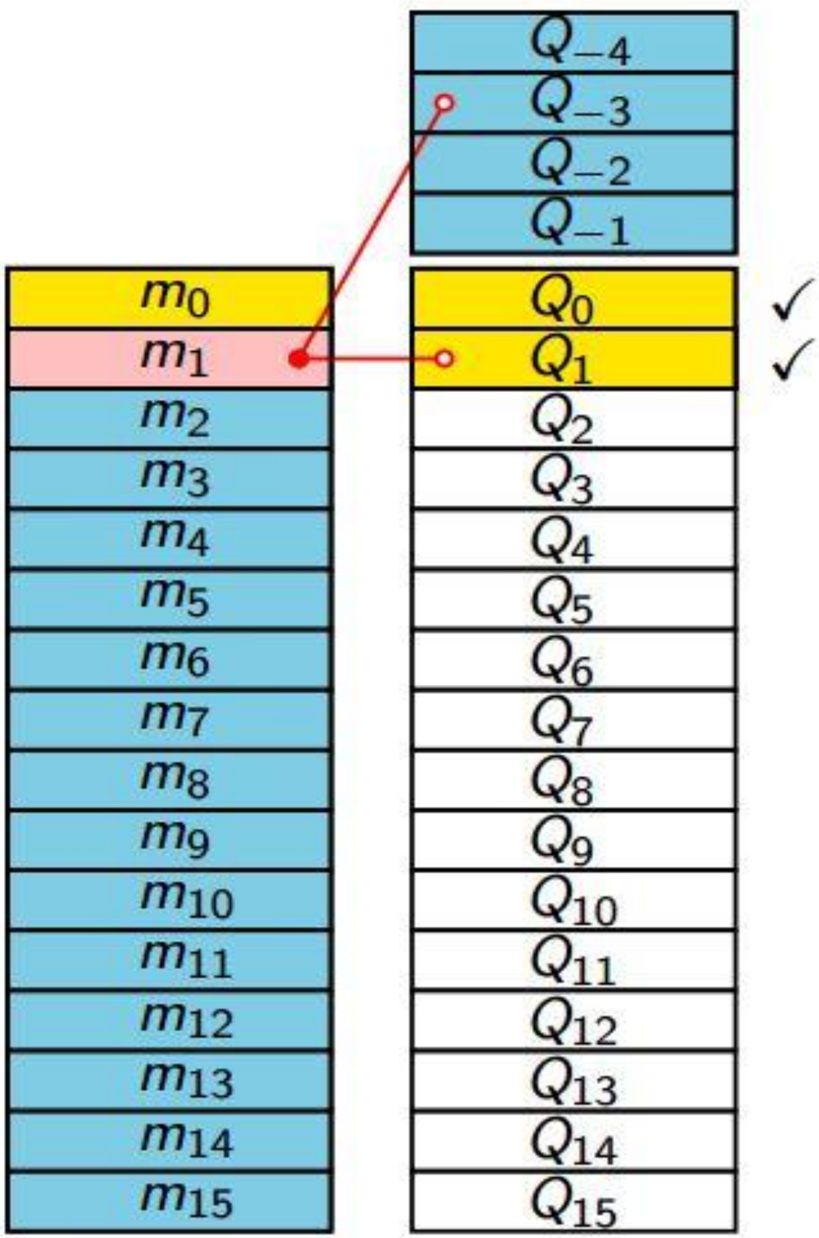
Wang's Approach to satisfy conditions in the first round



Message Modification

- *Select a message m_i*
- **Compute the corresponding Q_i**
- Modify Q_i to satisfy the conditions. Recompute m_i

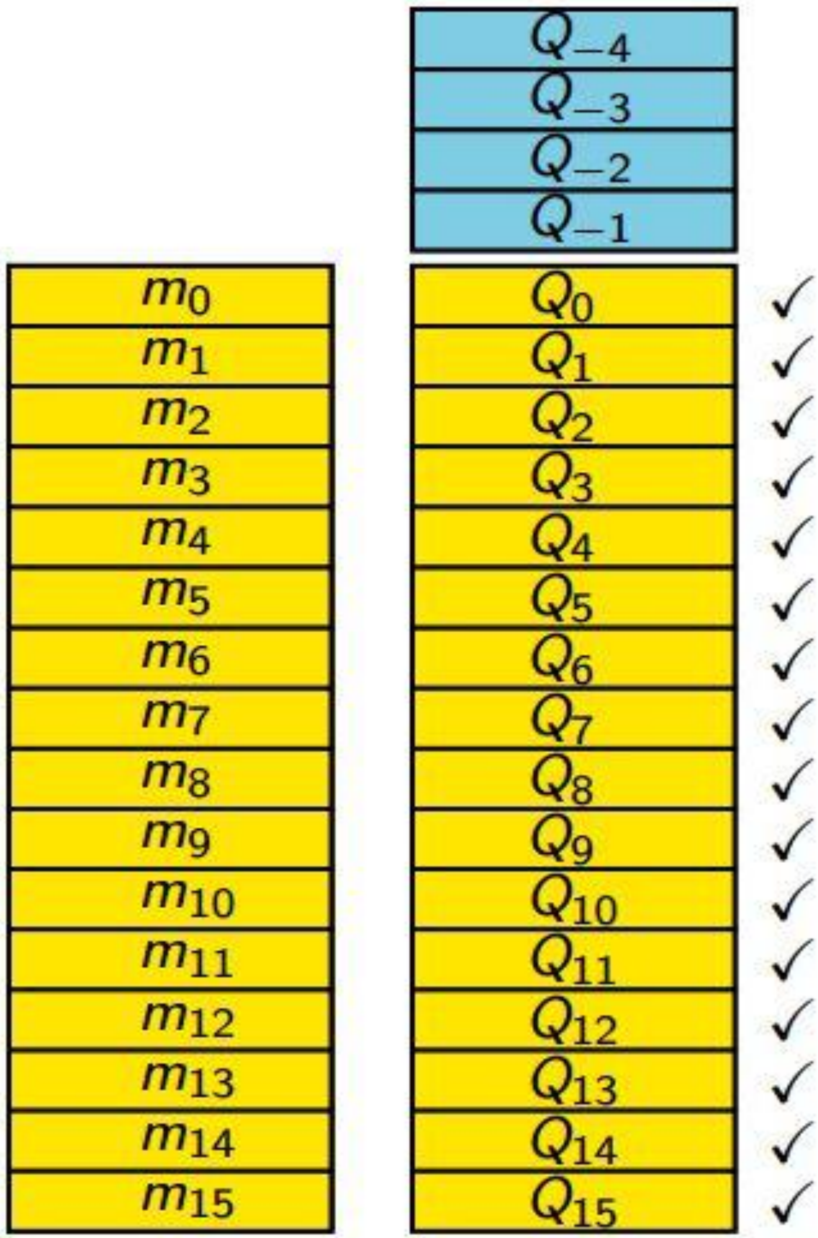
Wang's Approach to satisfy conditions in the first round



Message Modification

- *Select a message m_i*
- Compute the corresponding Q_i
- **Modify Q_i to satisfy the conditions. Recompute m_i**

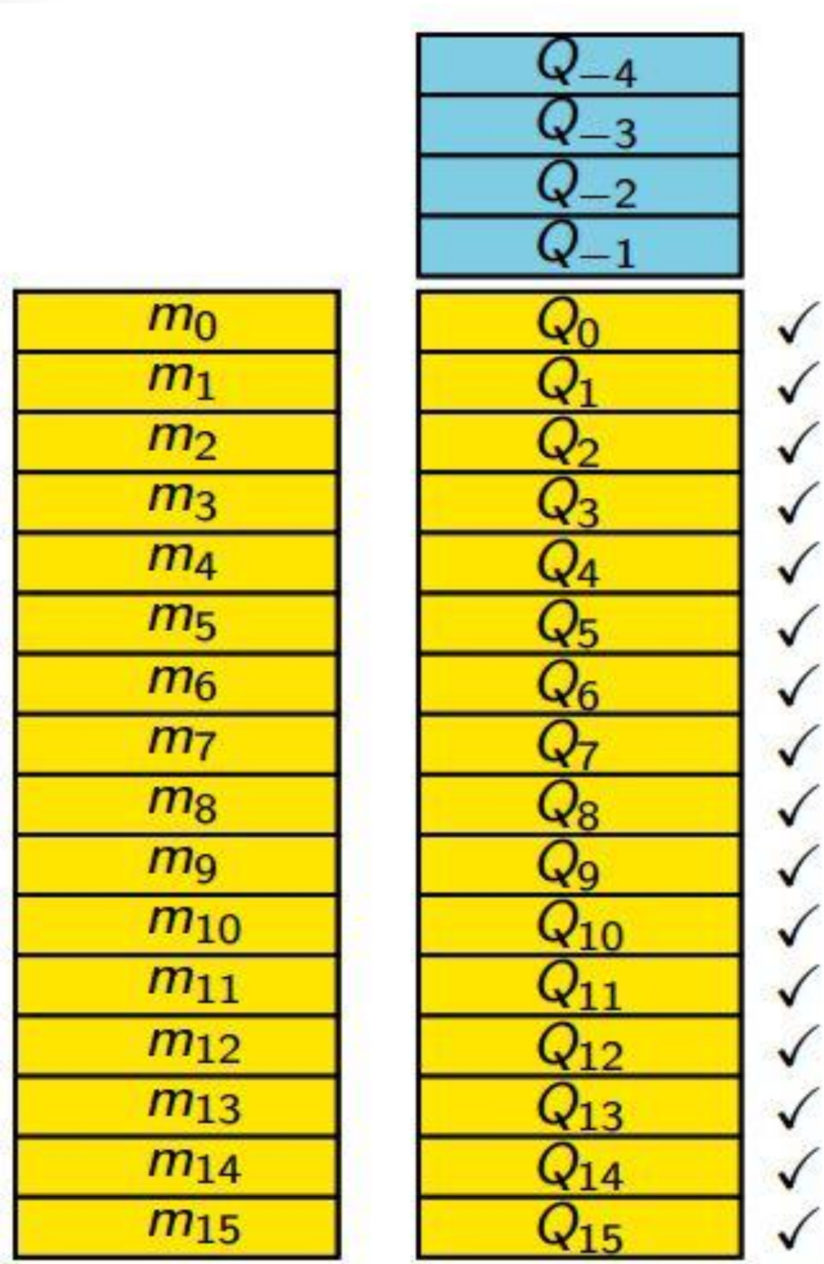
Wang's Approach to satisfy conditions in the first round

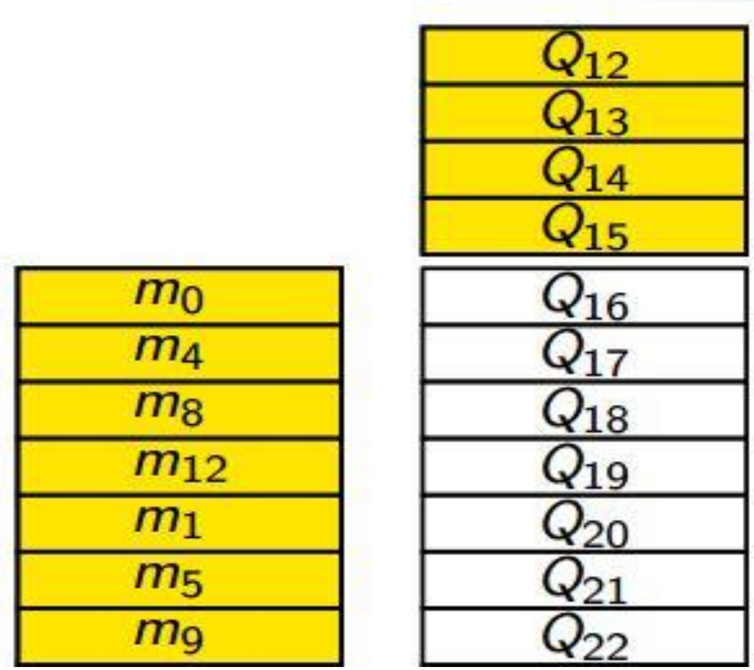


Message Modification

- *Select a message m_i*
- Compute the corresponding Q_i
- Modify Q_i to satisfy the conditions. Recompute m_i

Wang's Approach to satisfy conditions in the second round

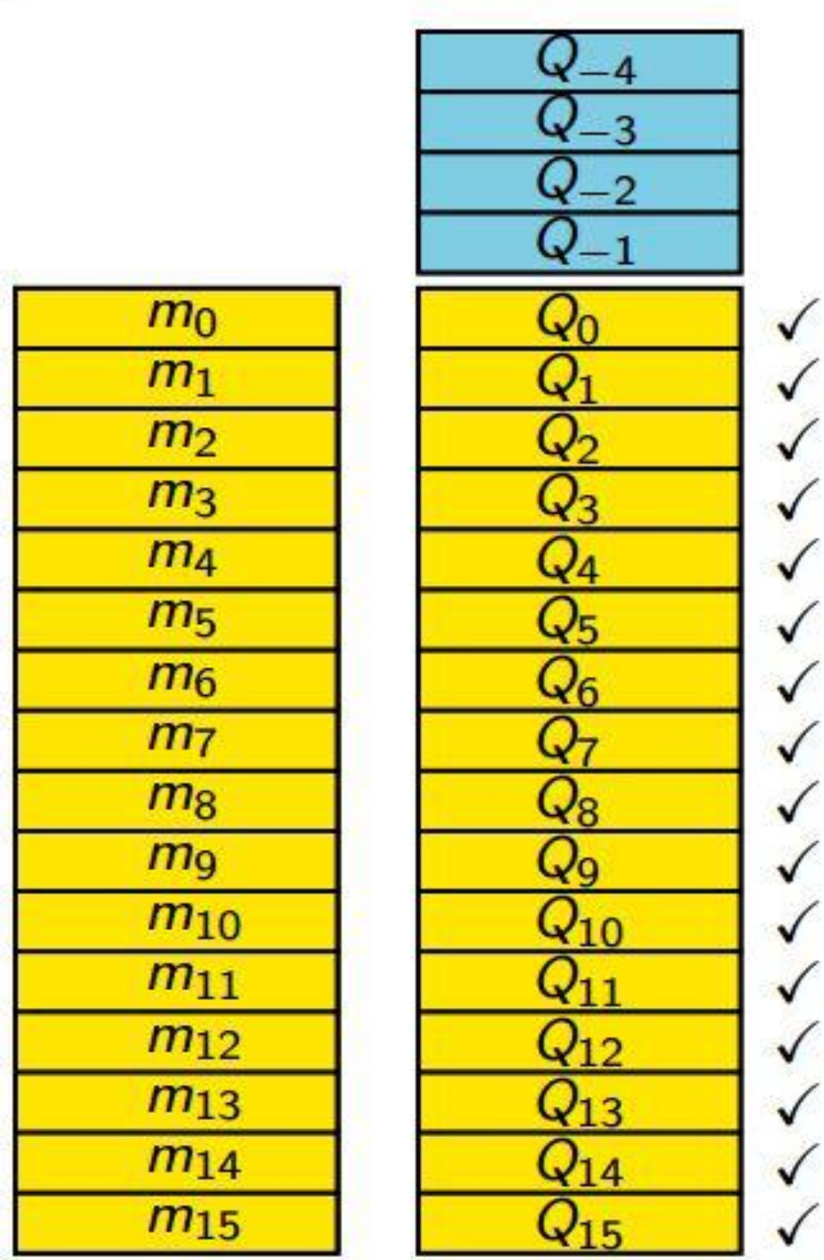


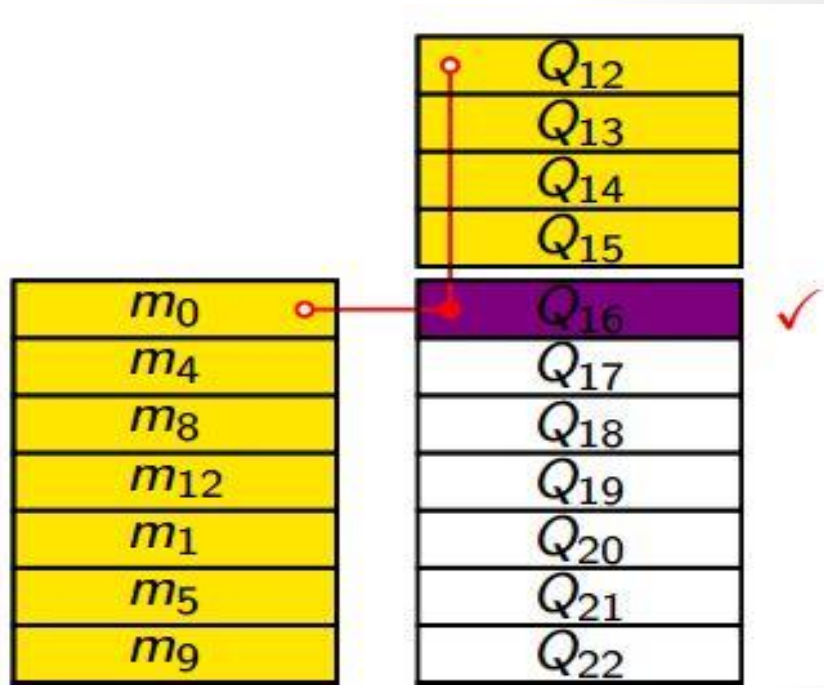


Multi Message Modification

- Compute Q_i.
- Modify Q_i and recompute m_i
- Recompute Q_i's and m_i's in the first round.

Wang’s Approach to satisfy conditions in the second round

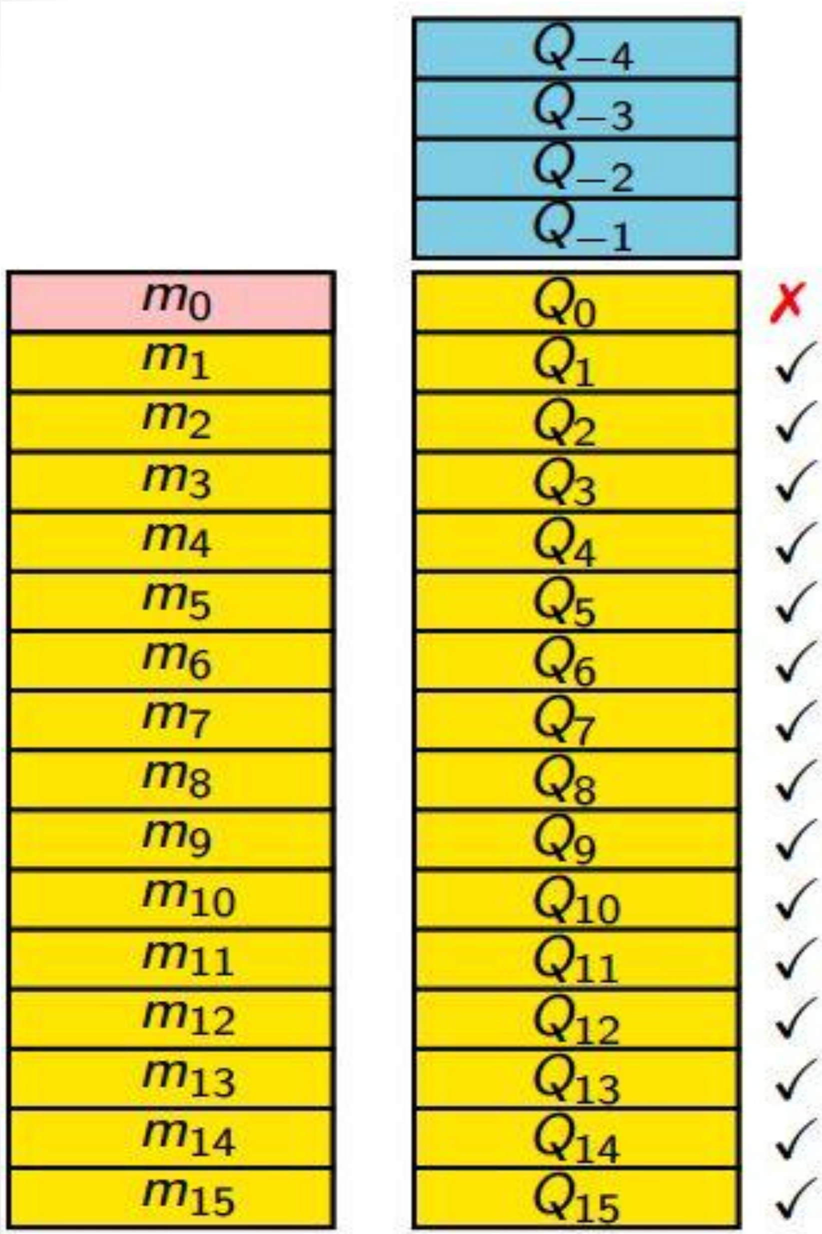


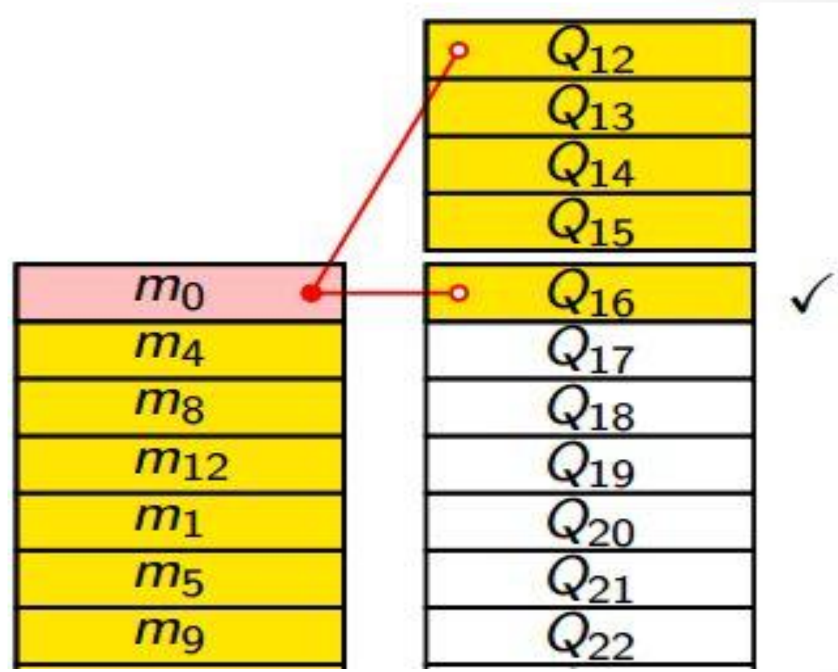


Multi Message Modification

- **Compute Q_i.**
- Modify Q_i and recompute m_i
- Recompute Q_i’s and m_i’s in the first round.

Wang's Approach to satisfy conditions in the second round

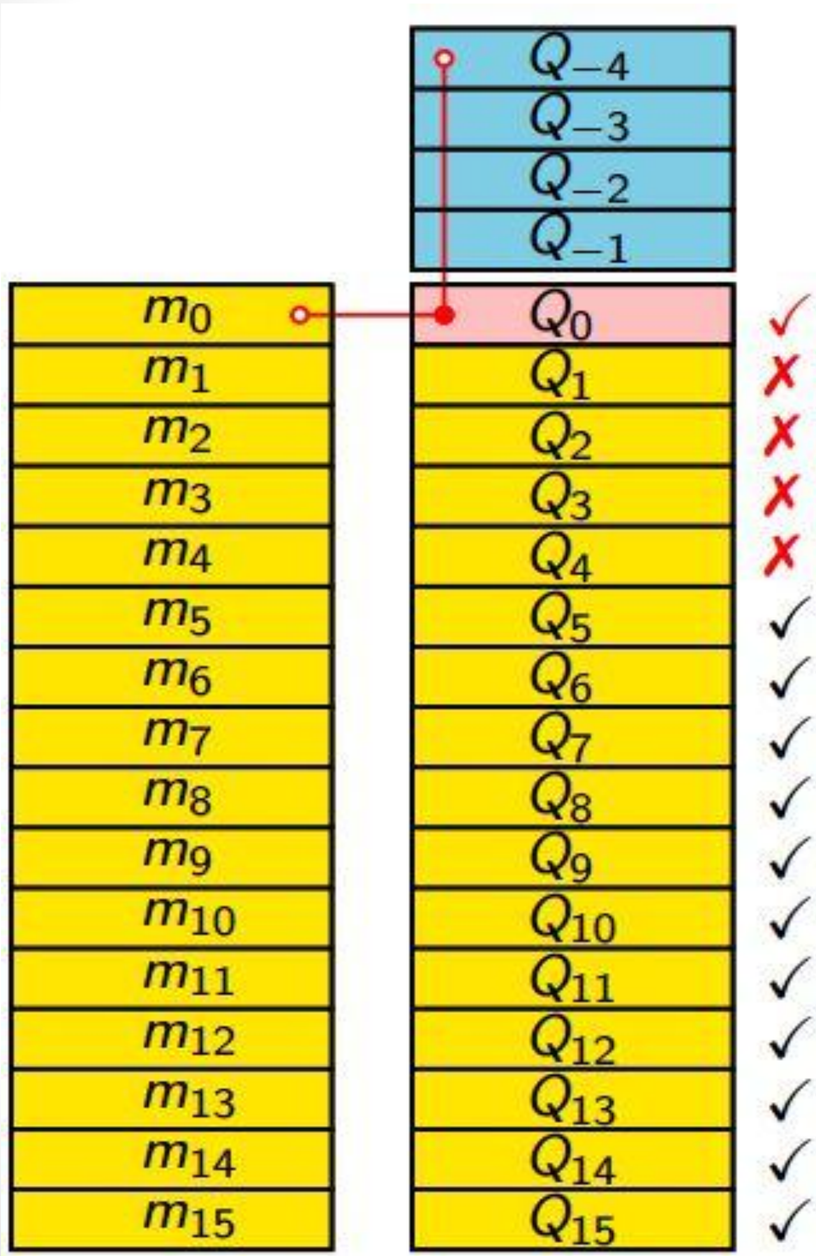


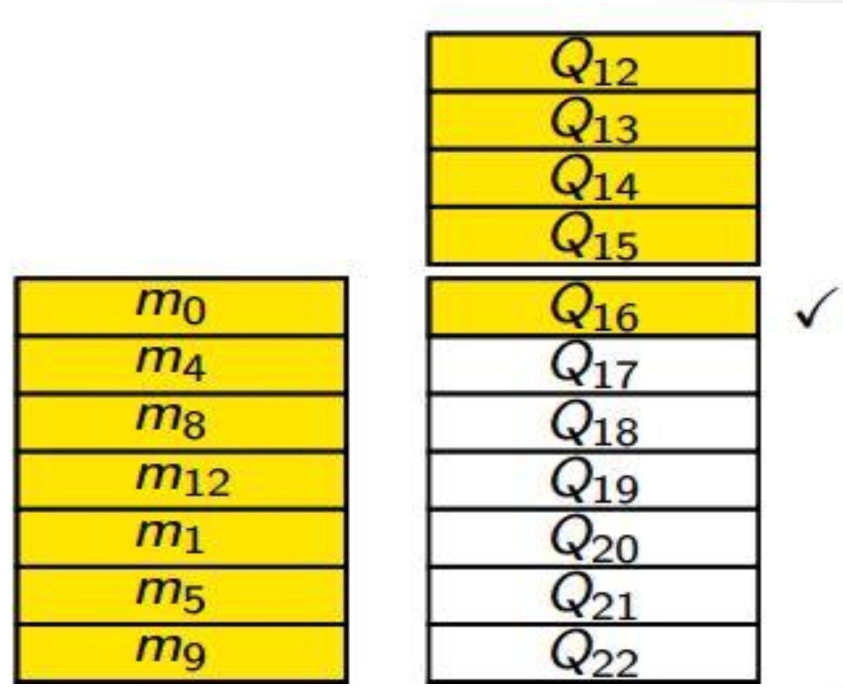


Multi Message Modification

- Compute Q_i.
- **Modify Q_i and recompute m_i**
- Recompute Q_i's and m_i's in the first round.

Wang’s Approach to satisfy conditions in the second round

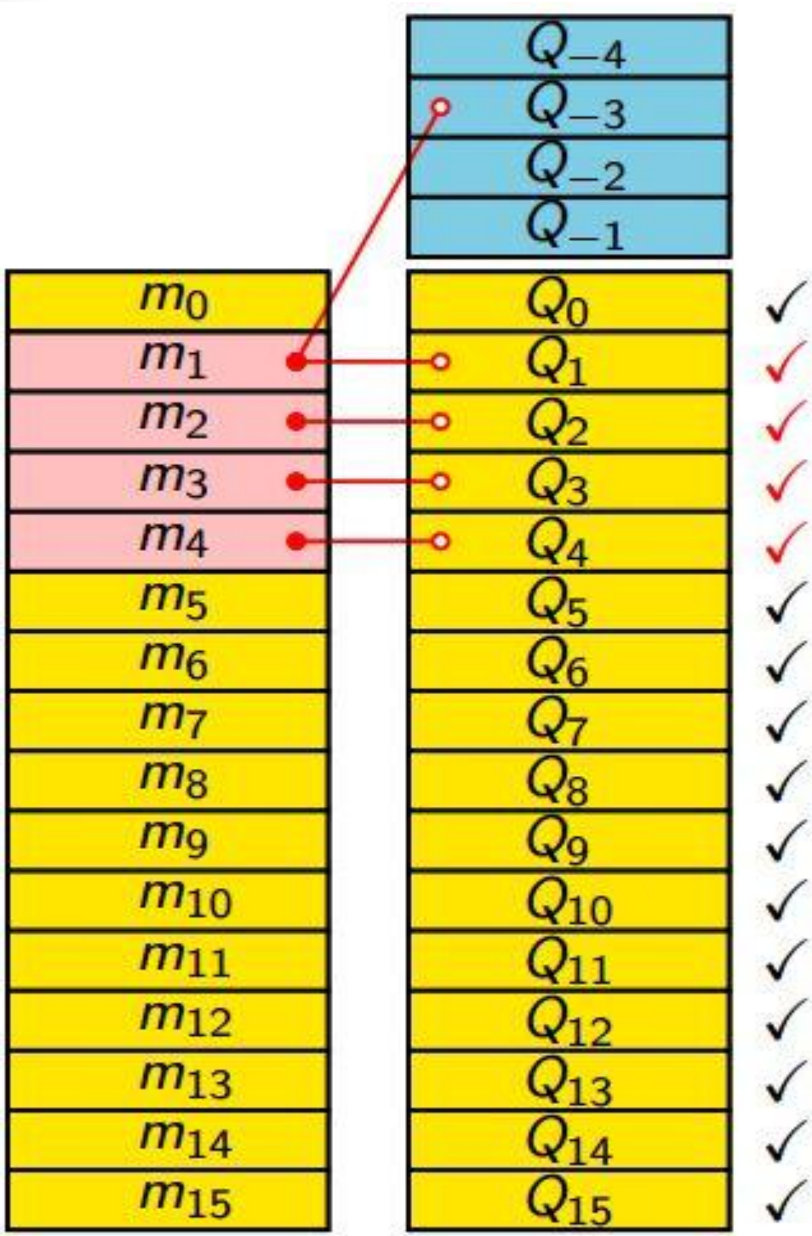


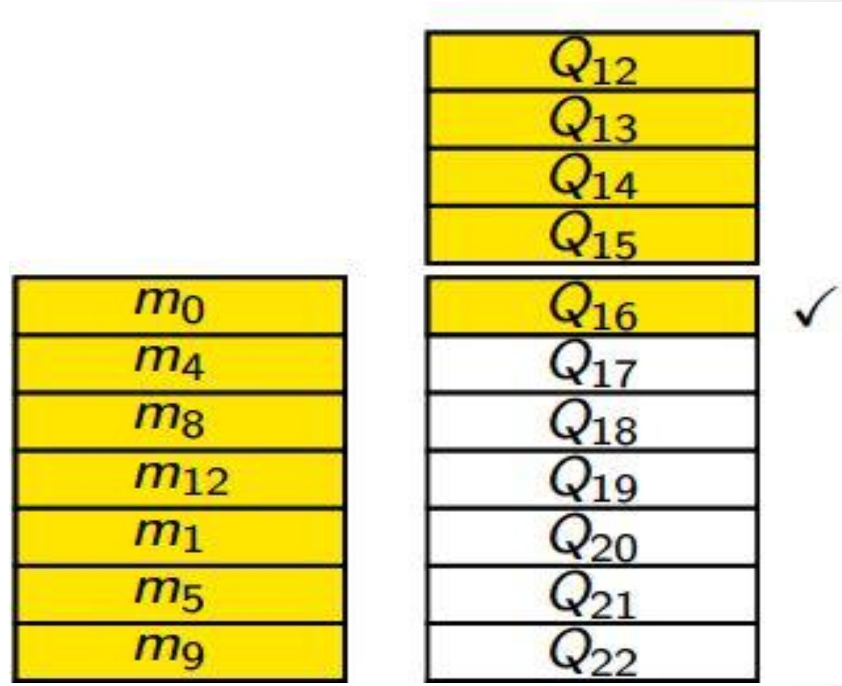


Multi Message Modification

- Compute Q_i.
- Modify Q_i and recompute m_i
- **Recompute Q_i’s** and m_i’s in the first round.

Wang’s Approach to satisfy conditions in the second round





Multi Message Modification

- Compute Q_i.
- Modify Q_i and recompute m_i
- Recompute Q_i’s and **m_i’s in the first round**.

Problem with Wang's Attack

- Due to the message modification technique, the colliding block cannot be chosen and look random.
- Hence there is no message freedom.
- Also since the exact approach of this attack is not yet known, it is not possible to establish an attack with a given message difference.

Algorithm By Gaetan Leurent –
A New Approach to Collision Finding

- We will assume that we are given the set of conditions on the internal state variables Q_i that produces collision.

- We will try to find a message M such that when one computes a hash of this message, the conditions on Q_i's hold.

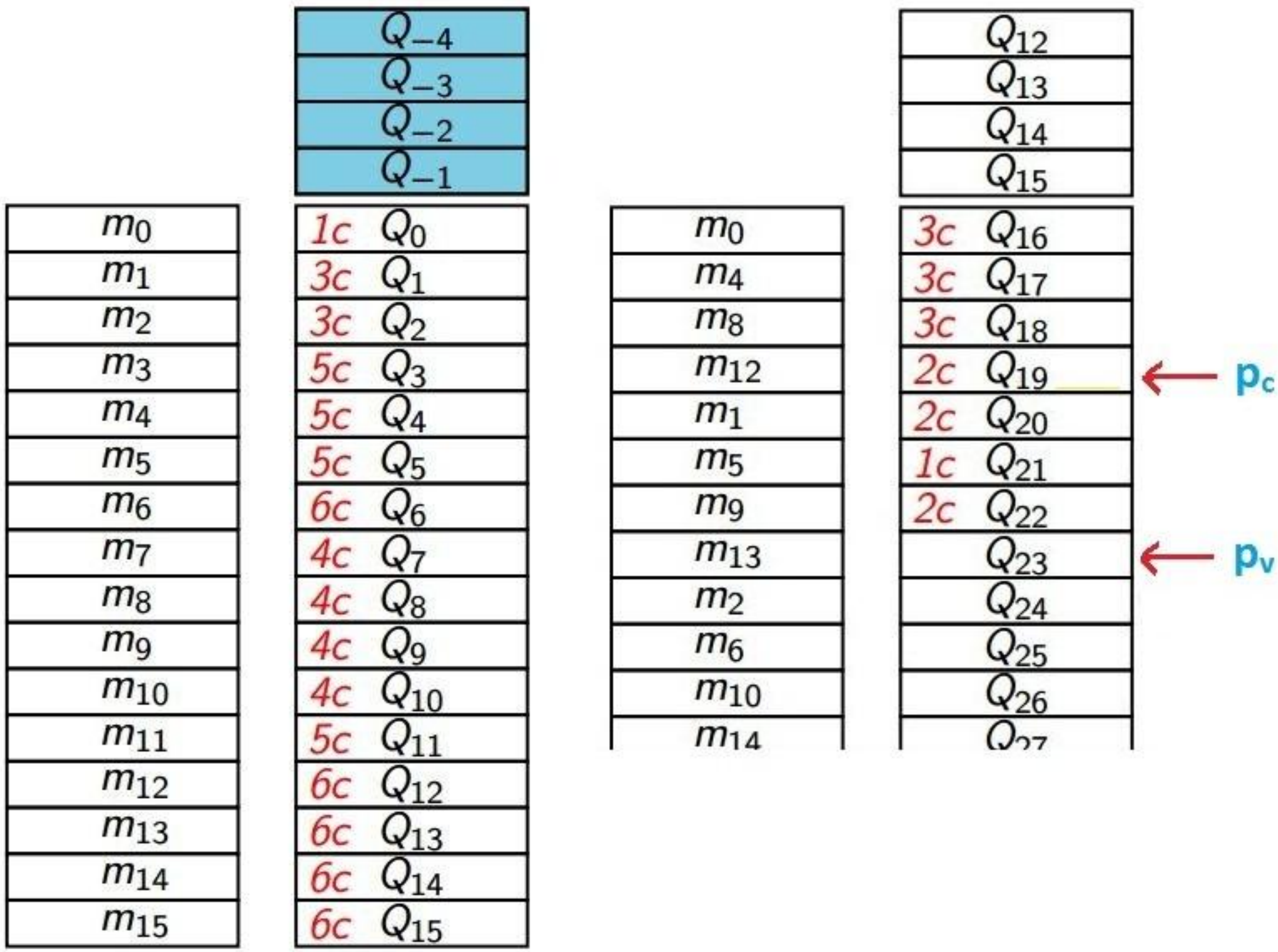
Tunnels

- Introduced by V. Klima in 2005.
- It speeds up the collision search
- Point of verification (p_v) is the step where we will start using tunnels.
- Point of choice (p_c) is the first step whose conditions will not be satisfied deterministically.
- A tunnel is a message modification technique that does not affect the conditions upto p_v-1 (point of verification).

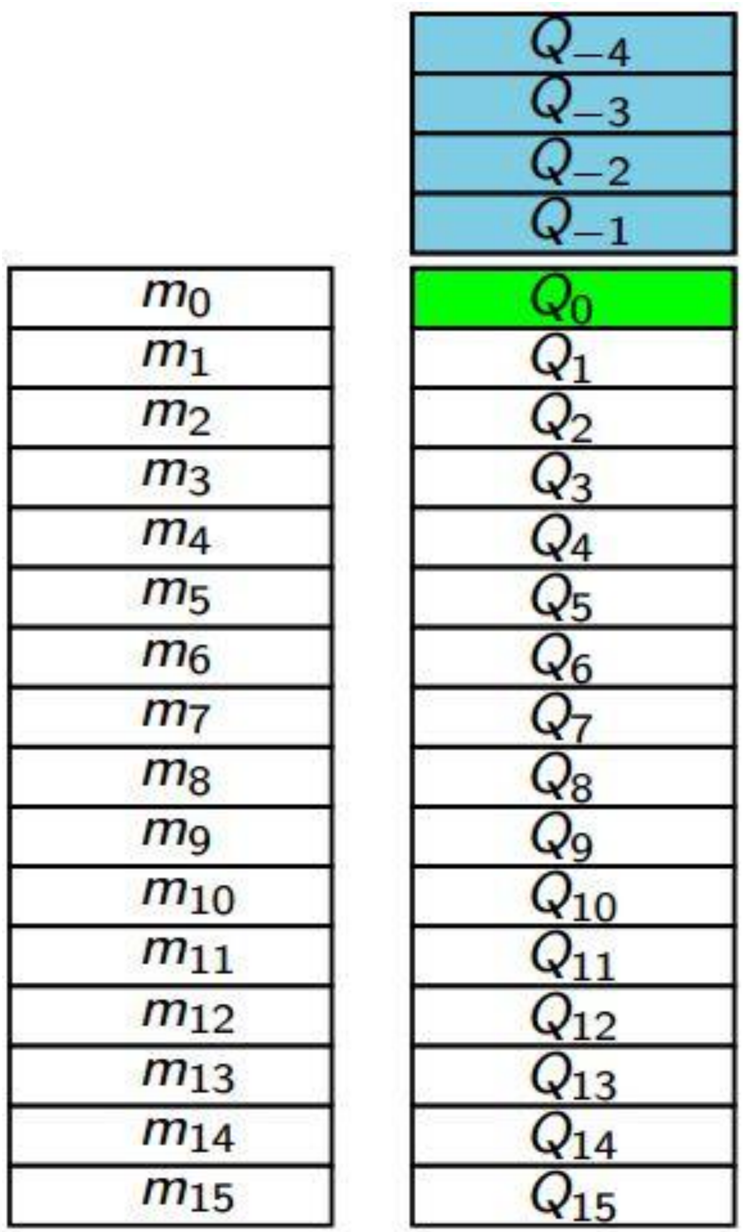
The Method

- We will start fixing Q_i from the middle. (will allow us to deal with the first round and the beginning of the second round simultaneously)
- We will choose the Q_i's till the step p_c.
- We will compute the Q_i's from the previous Q_i's for the steps p_c to p_v.
- Using the tunnels, we will try all possible messages that satisfies all the conditions from p_v till the end of the round.

p_c and p_v in MD4



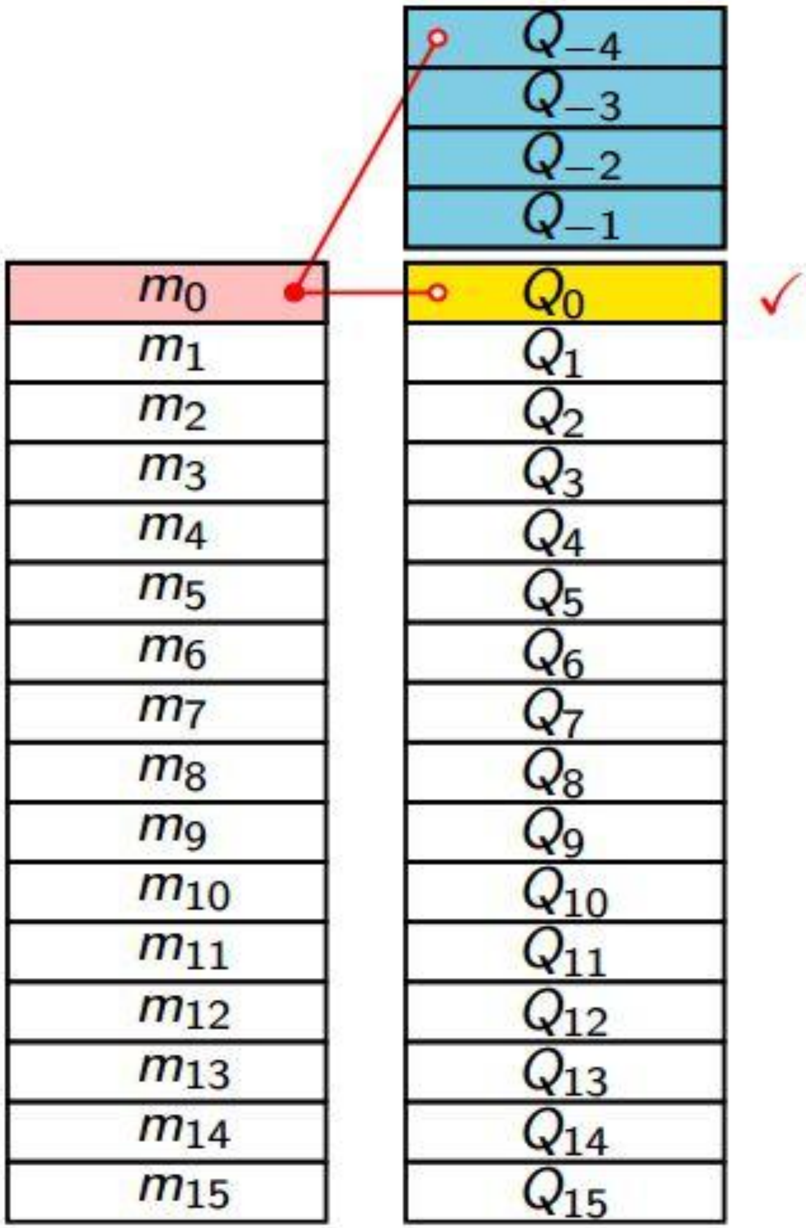
Approach to satisfy condition in the first round



Approach

- **Choose Q_i**
- Choose m_i

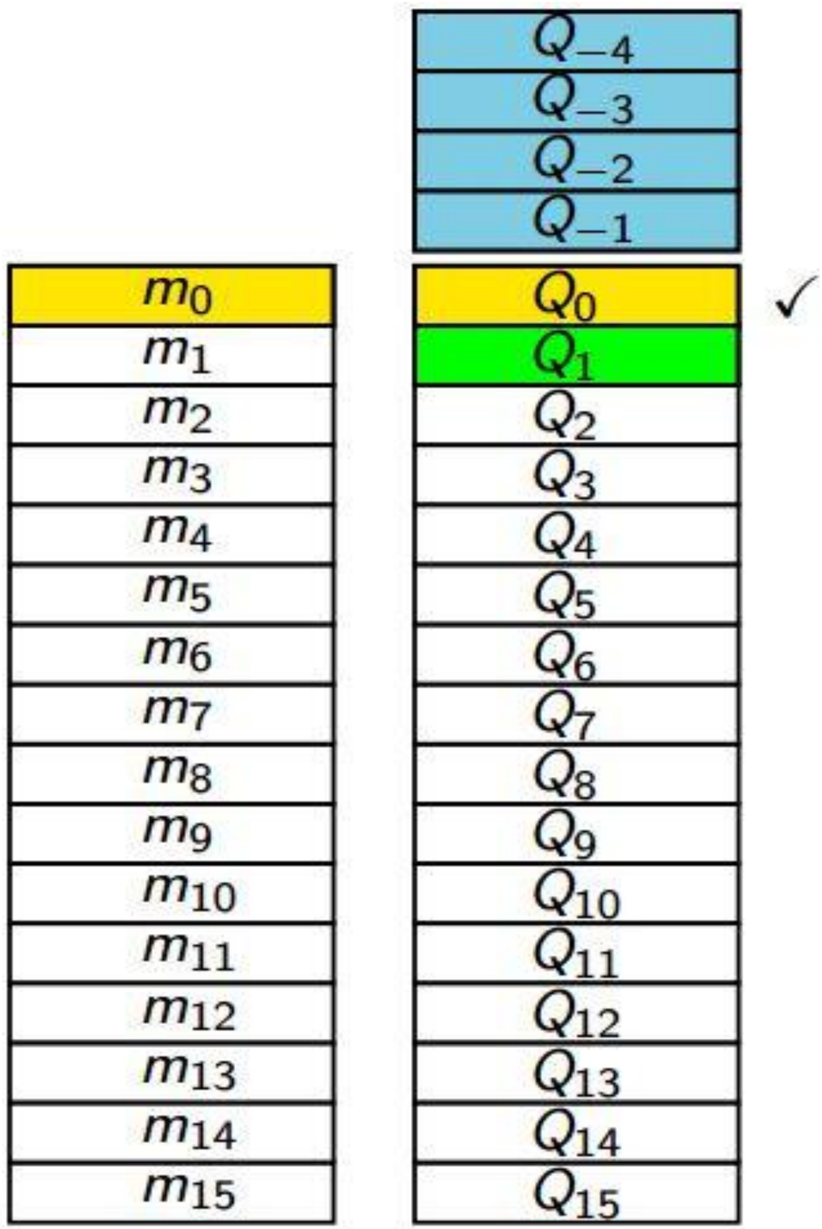
Approach to satisfy condition in the first round



Approach

- Choose Q_i
- **Compute** m_i

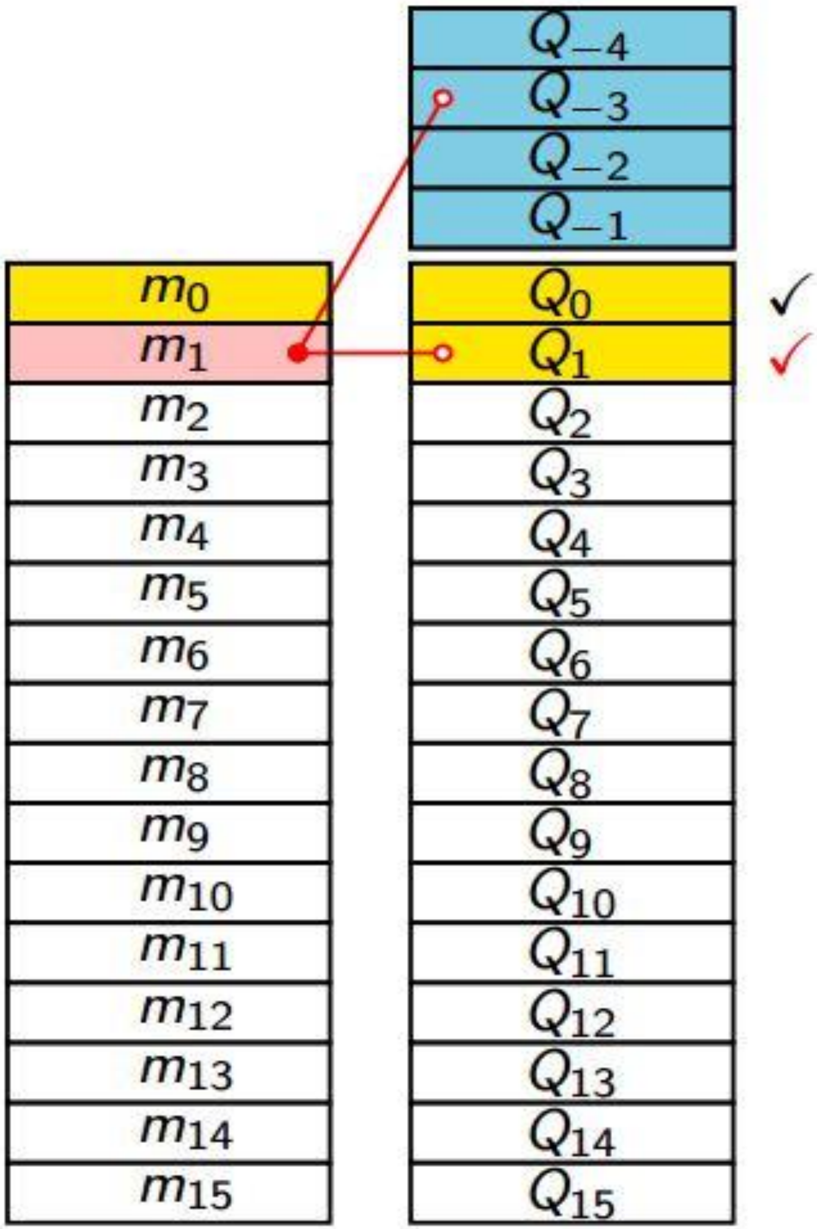
Approach to satisfy condition in the first round



Approach

- **Choose Q_i**
- Compute m_i

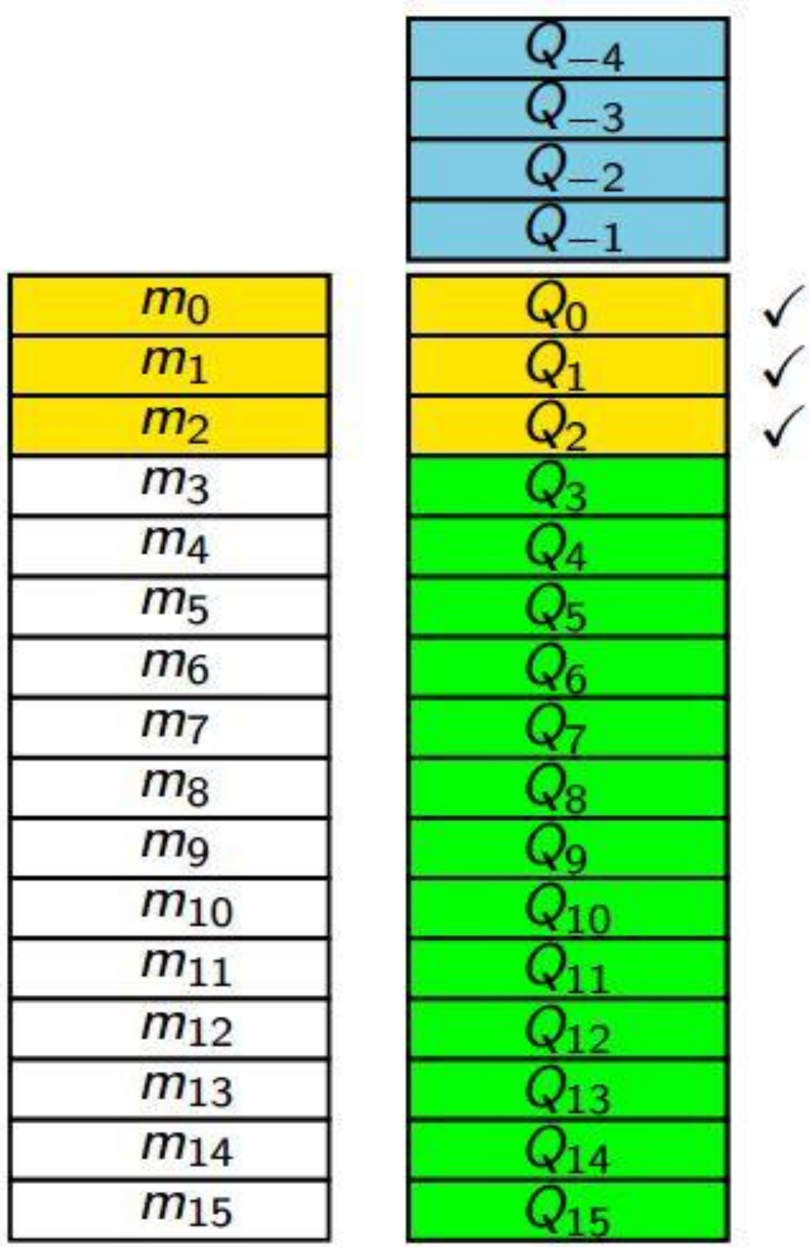
Approach to satisfy condition in the first round



Approach

- Choose Q_i
- **Compute m_i**

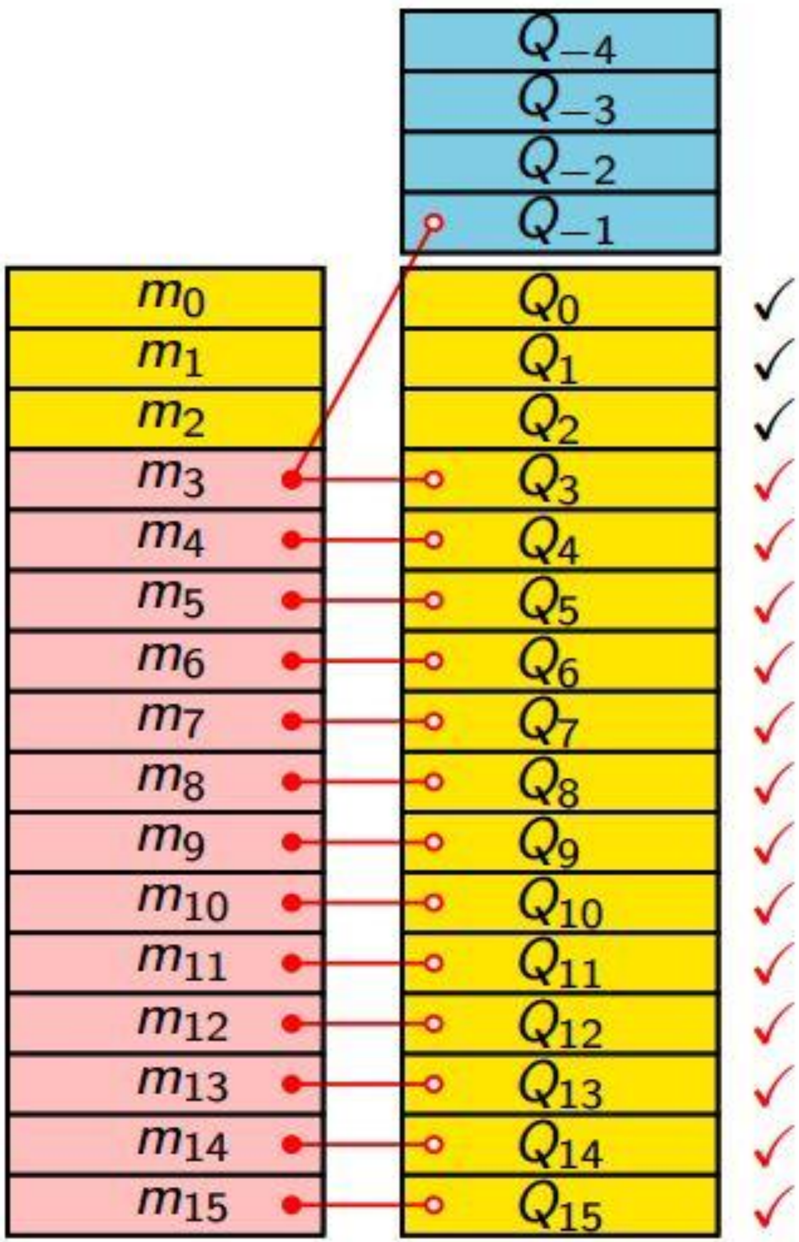
Approach to satisfy condition in the first round



Approach

- **Choose Q_i**
- Compute m_i

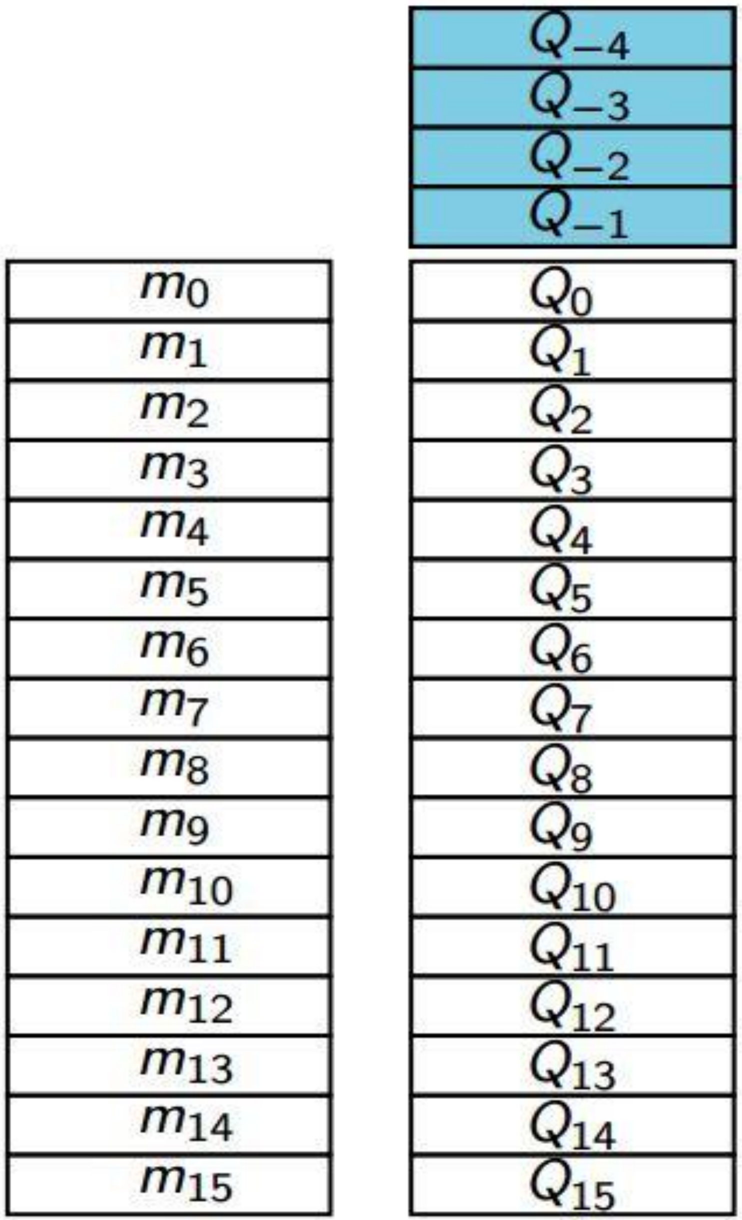
Approach to satisfy condition in the first round

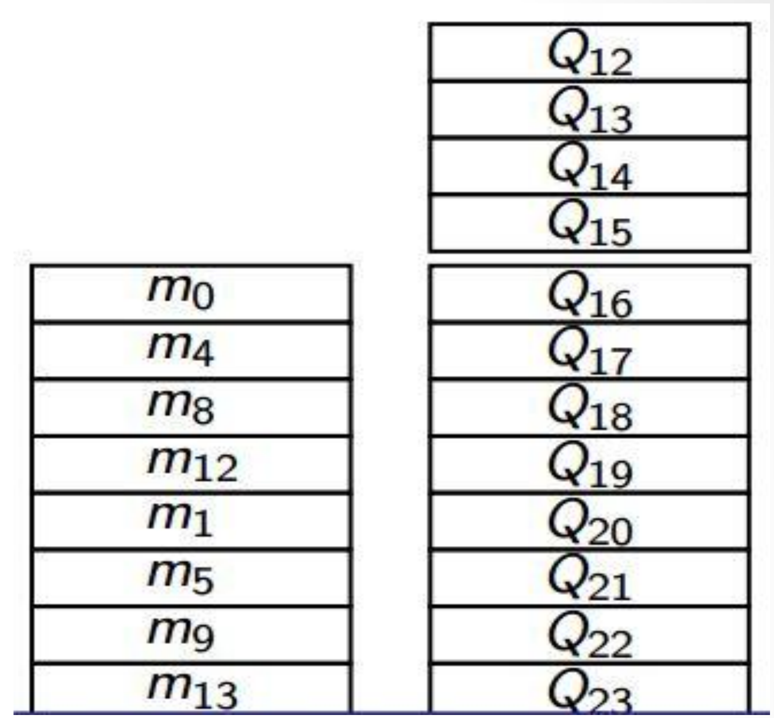


Approach

- Choose Q_i
- **Compute m_i**

Approach to satisfy condition in the second round

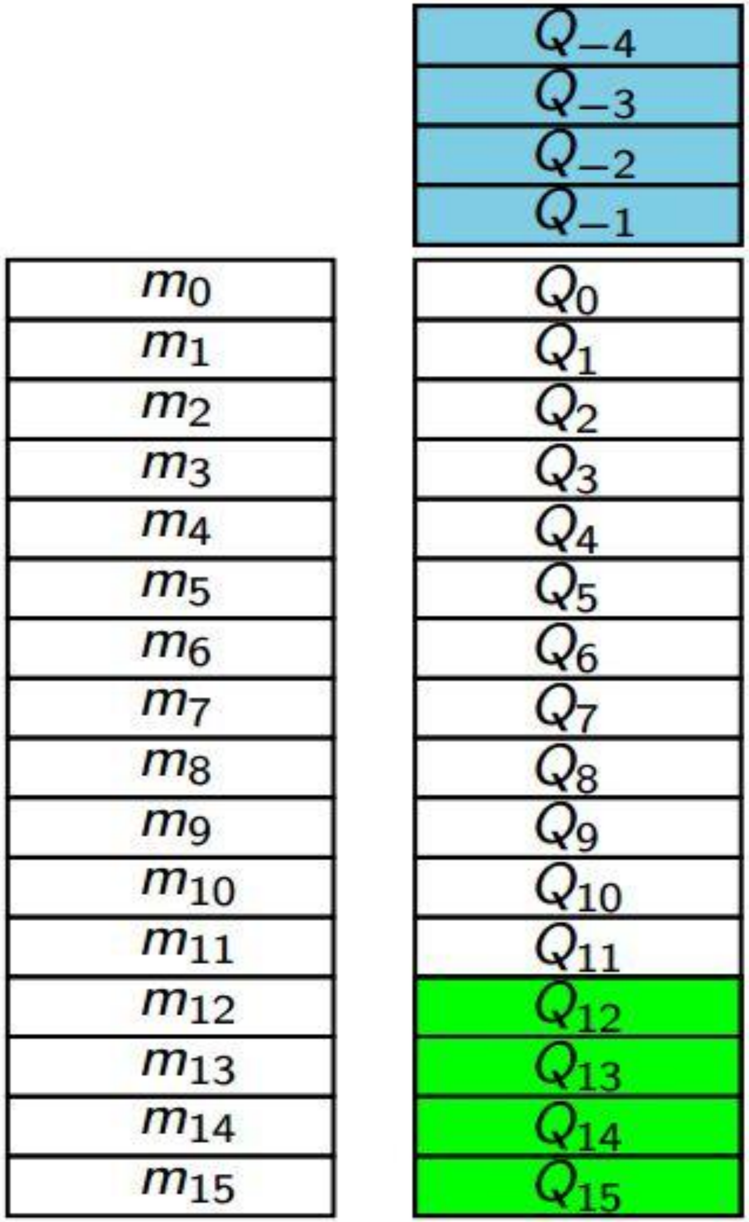


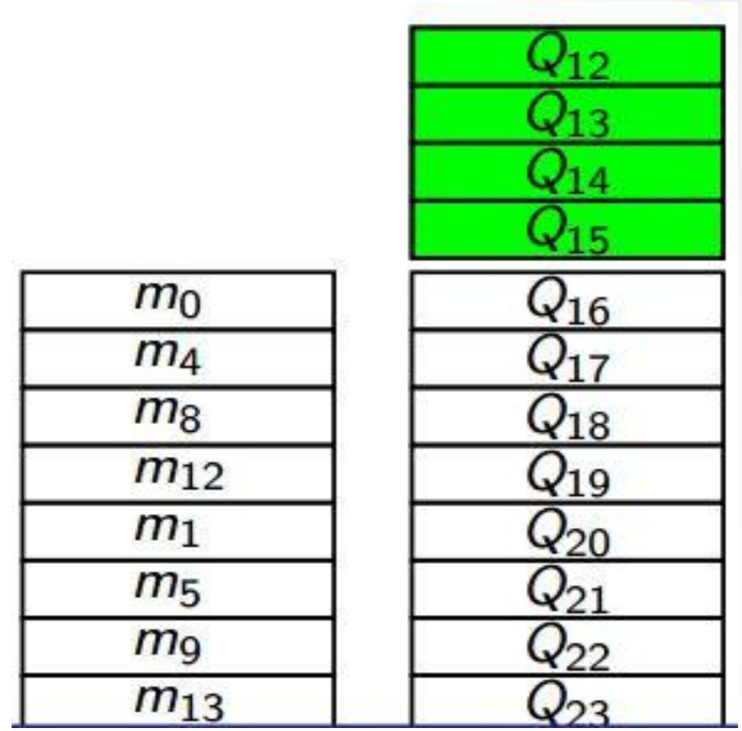


New Method

- Choose the end of the first round.
- Choose m_i to satisfy both the conditions.
- Fill the first round.

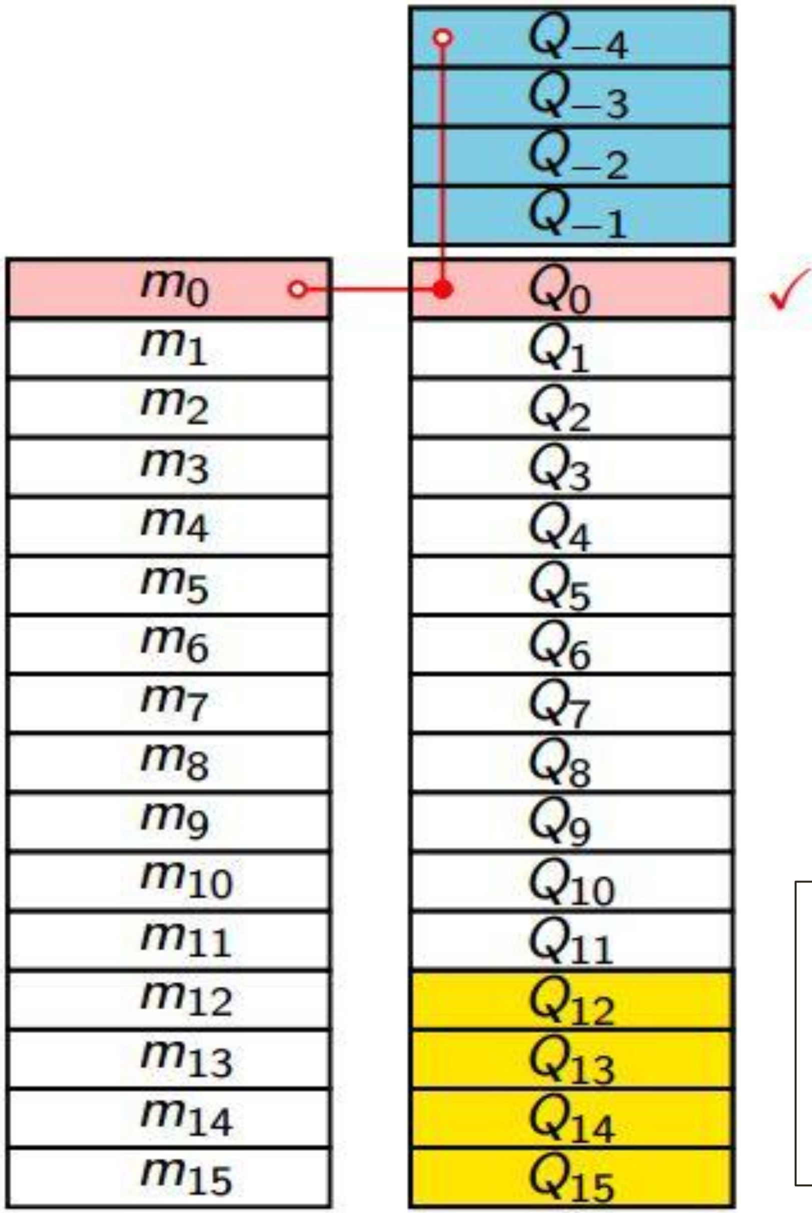
Approach to satisfy condition in the second round

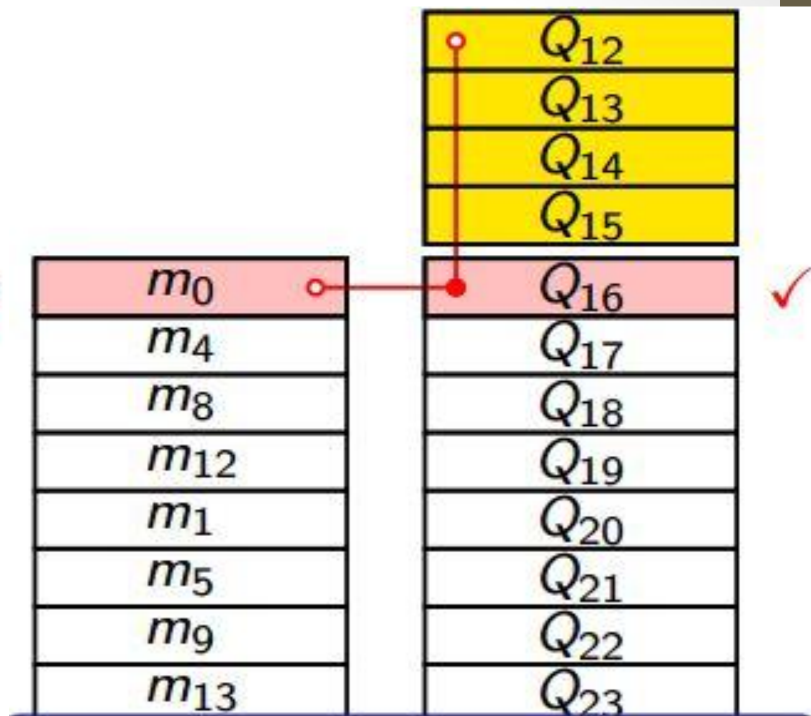




- **Choose the end of the first round**.
- Choose m_i to satisfy both the conditions.
- Fill the first round.

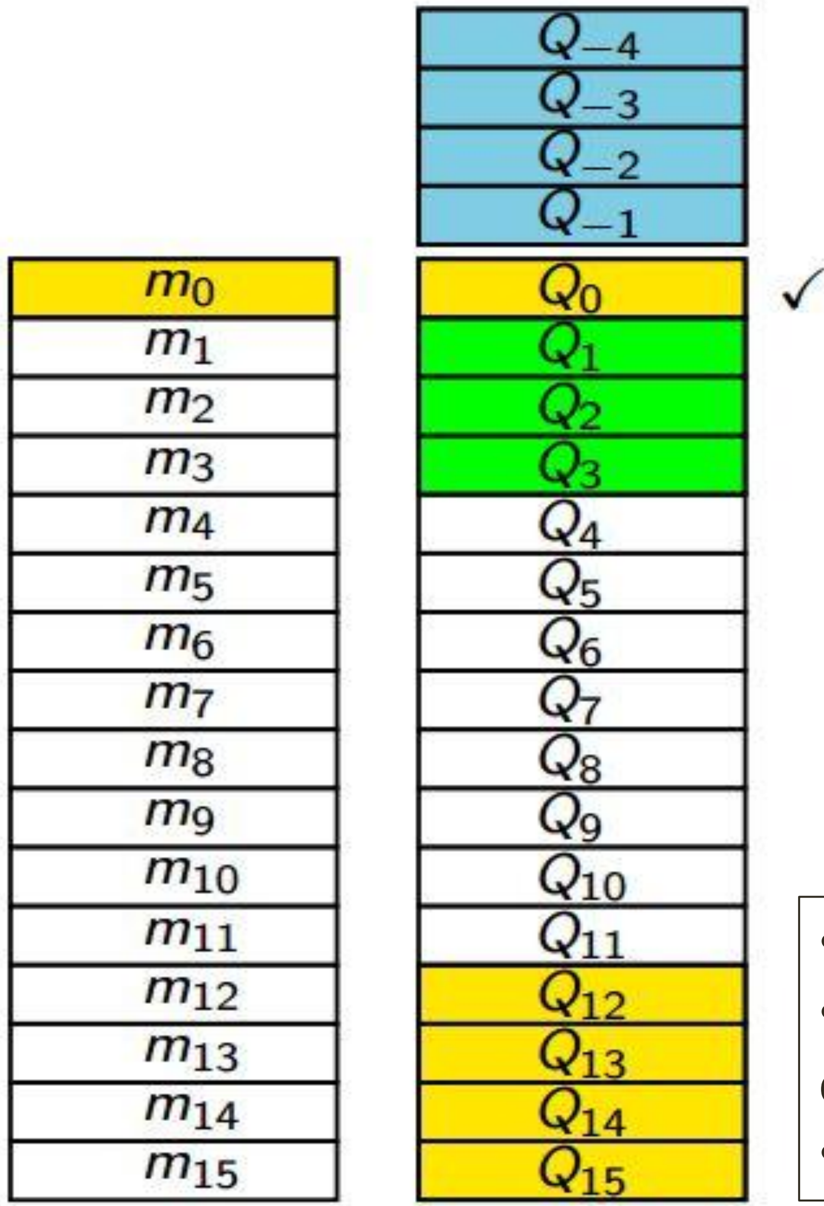
Approach to satisfy condition in the second round

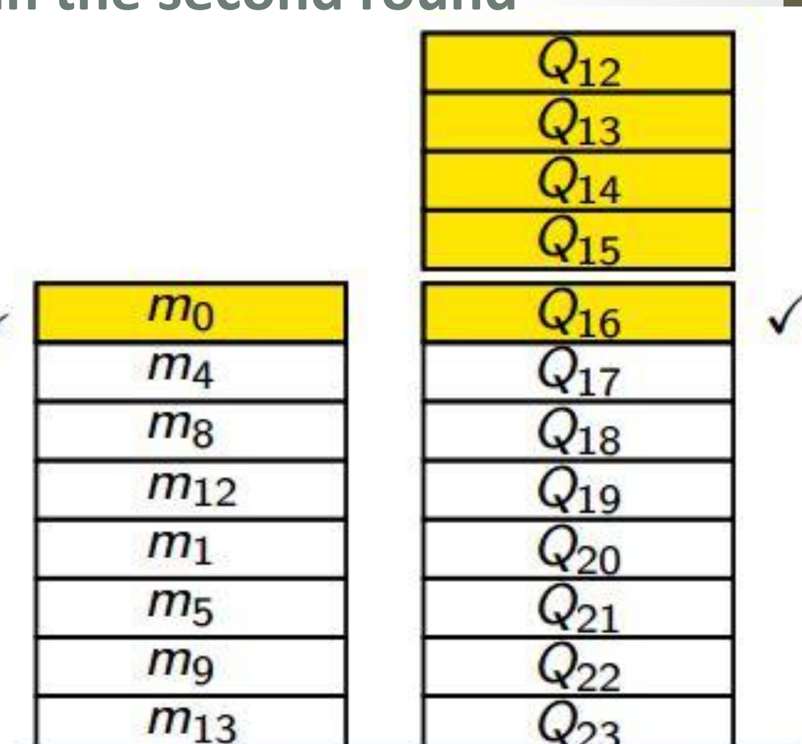




- Choose the end of the first round.
- **Choose m_i to satisfy both the conditions**.
- Fill the first round.

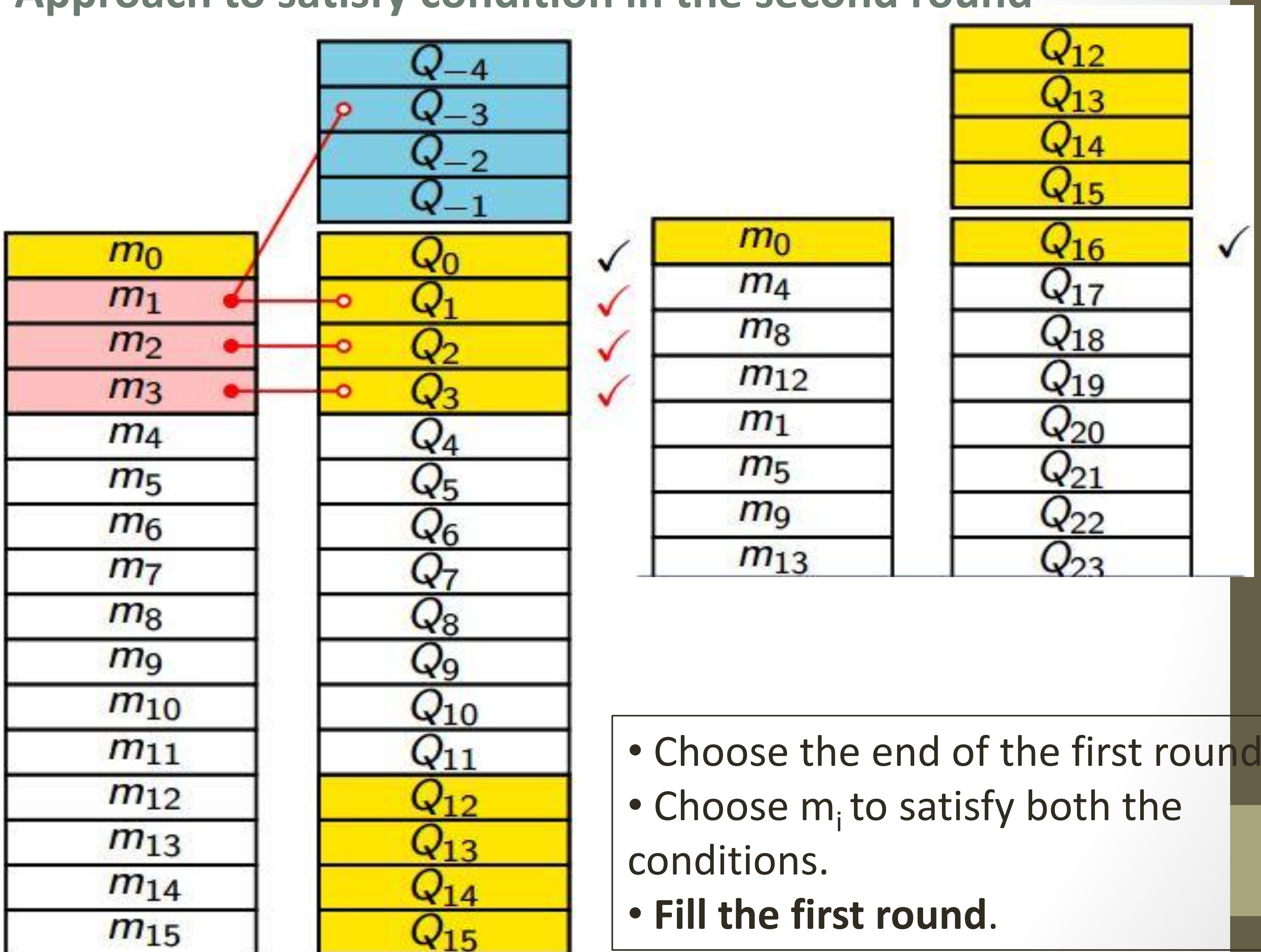
Approach to satisfy condition in the second round





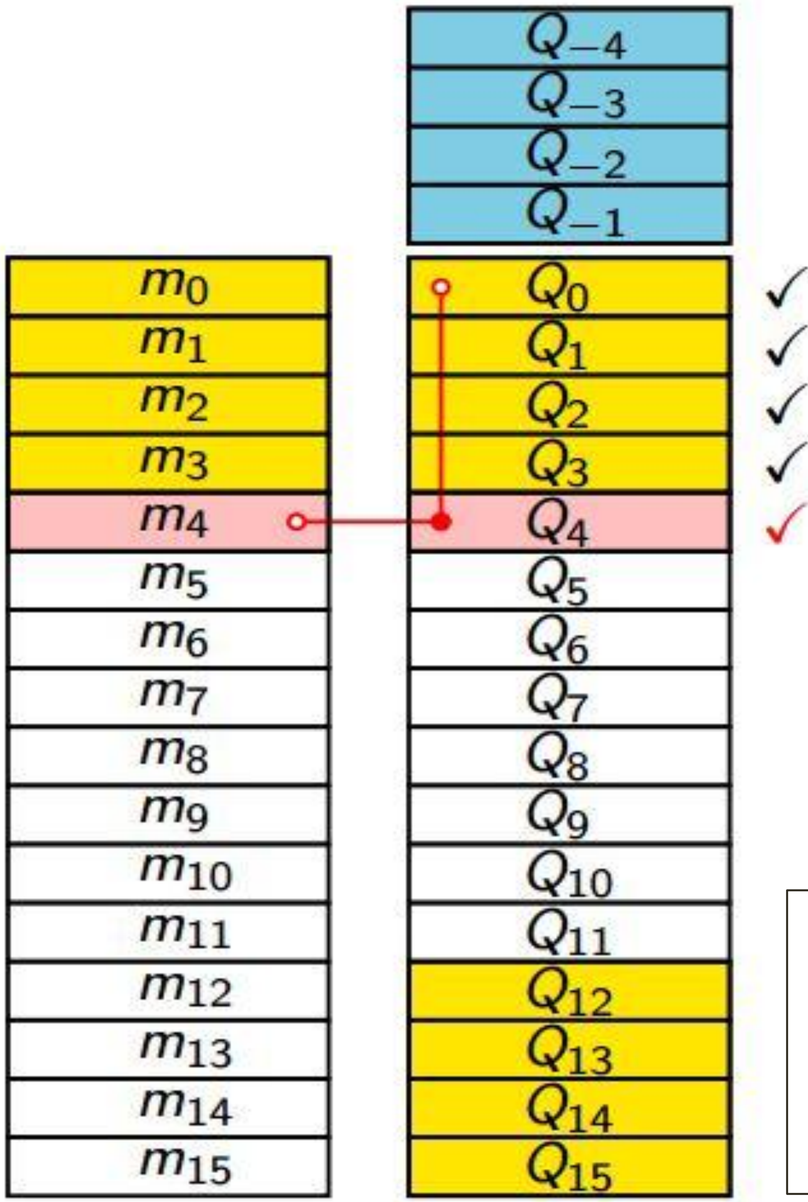
- Choose the end of the first round.
- Choose m_i to satisfy both the conditions.
- **Fill the first round.**

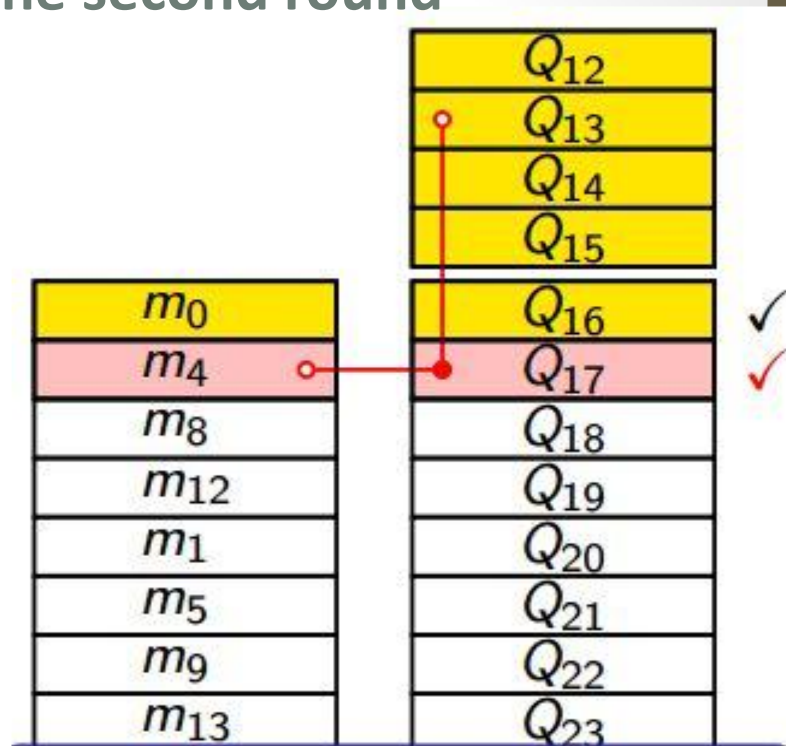
Approach to satisfy condition in the second round



- Choose the end of the first round.
- Choose m_i to satisfy both the conditions.
- **Fill the first round**.

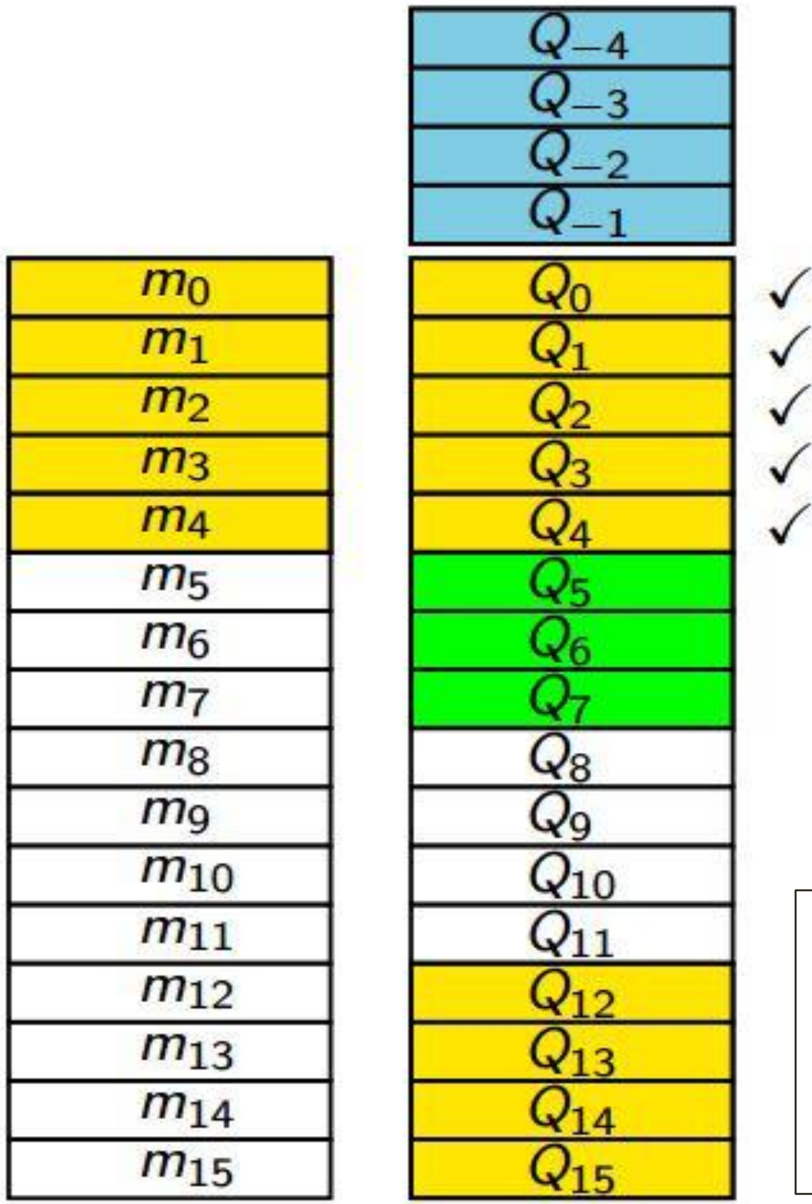
Approach to satisfy condition in the second round

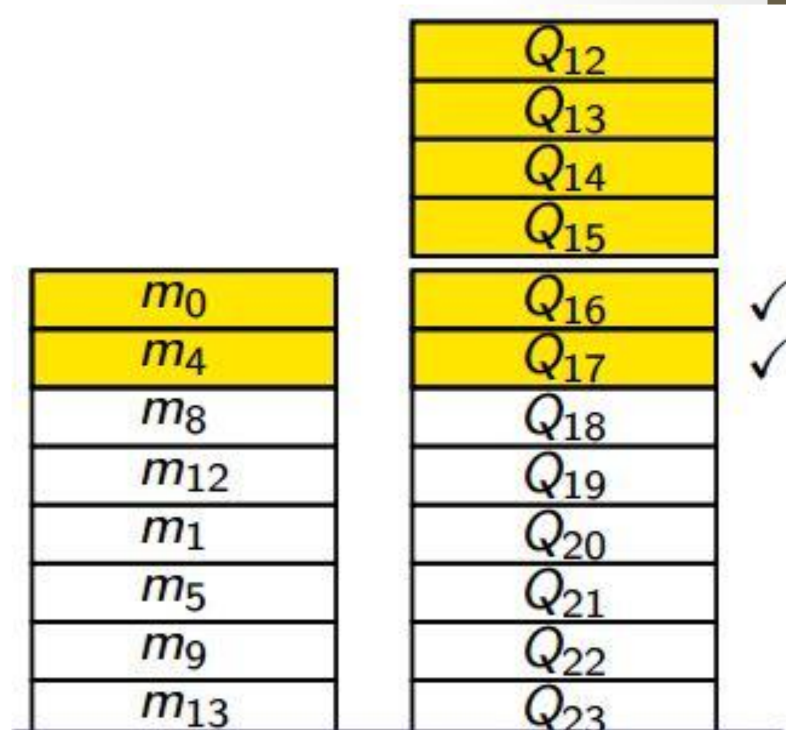




- Choose the end of the first round.
- **Choose m_i to satisfy both the conditions**.
- Fill the first round.

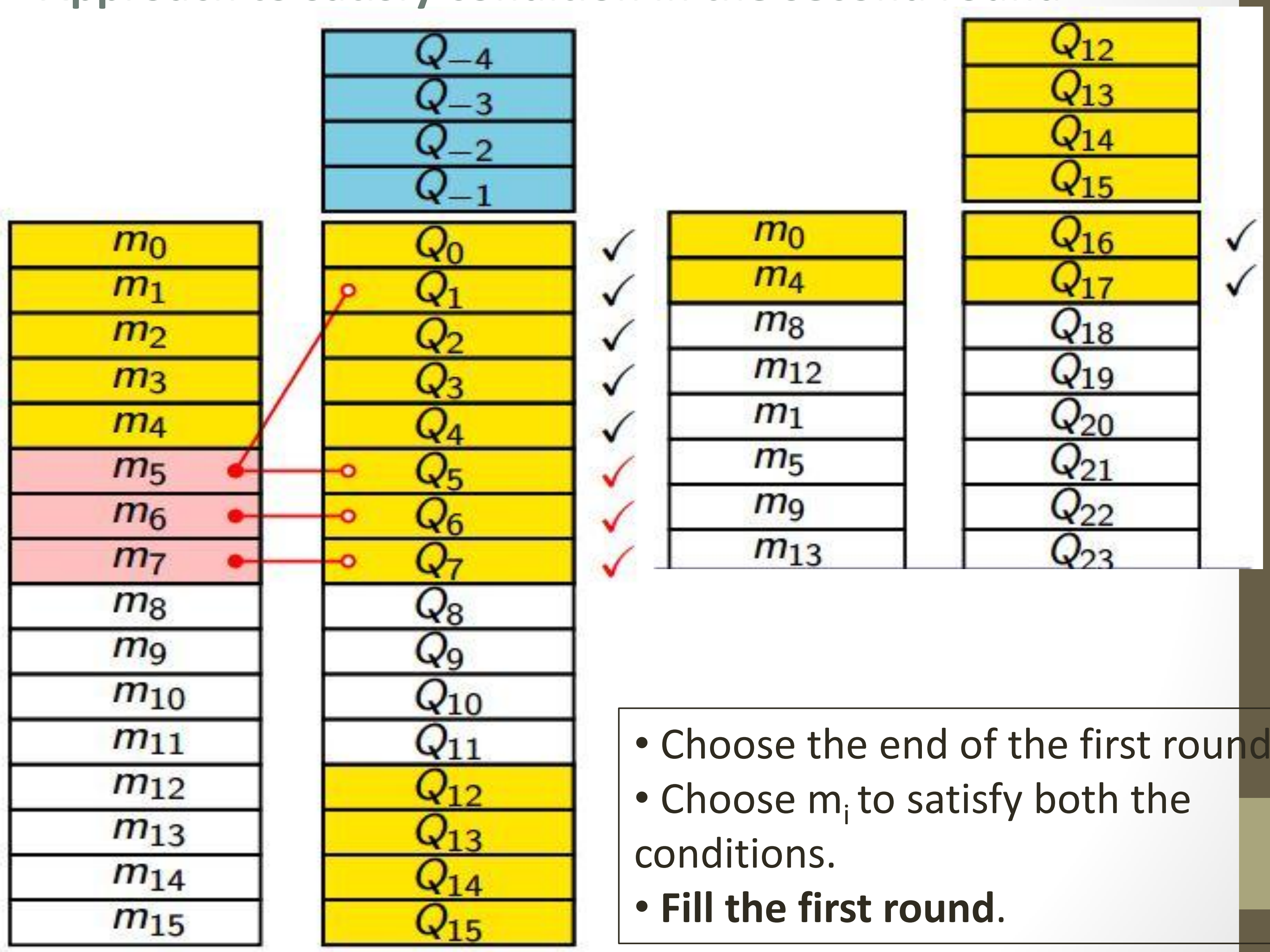
Approach to satisfy condition in the second round





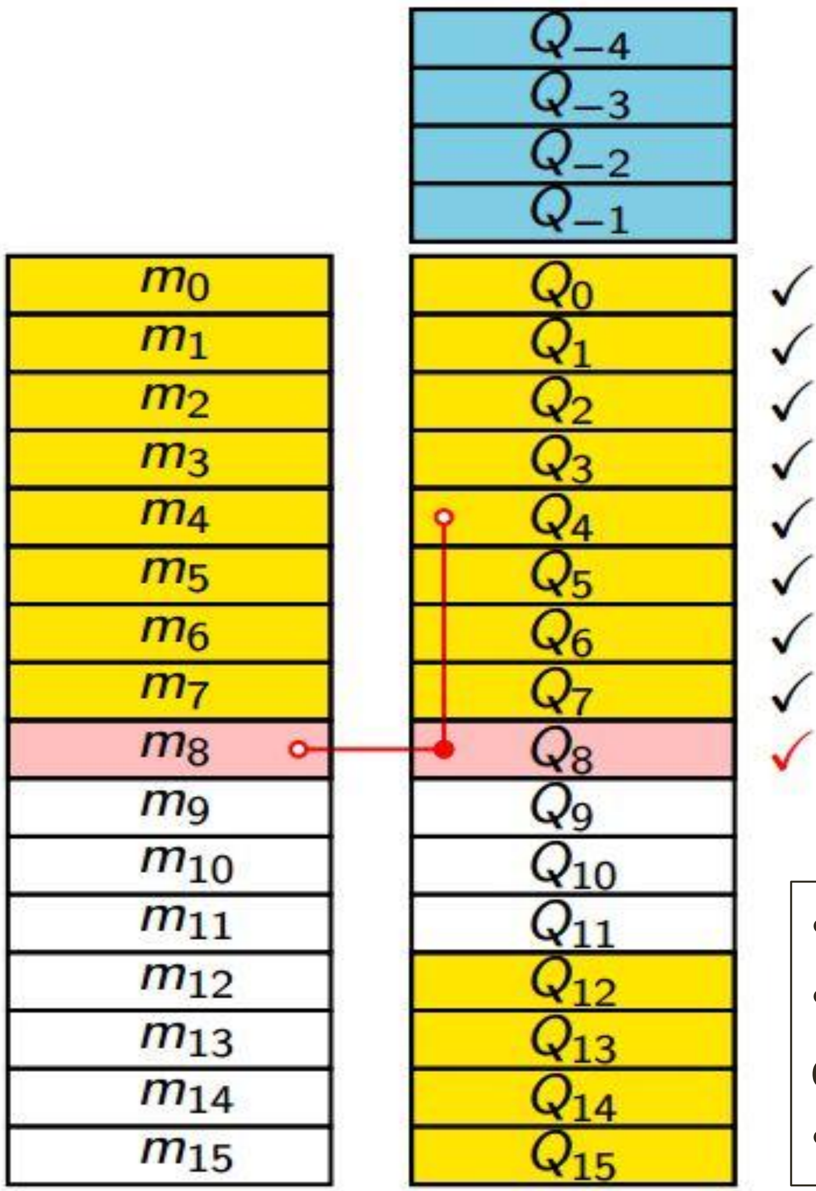
- Choose the end of the first round.
- Choose m_i to satisfy both the conditions.
- **Fill the first round**.

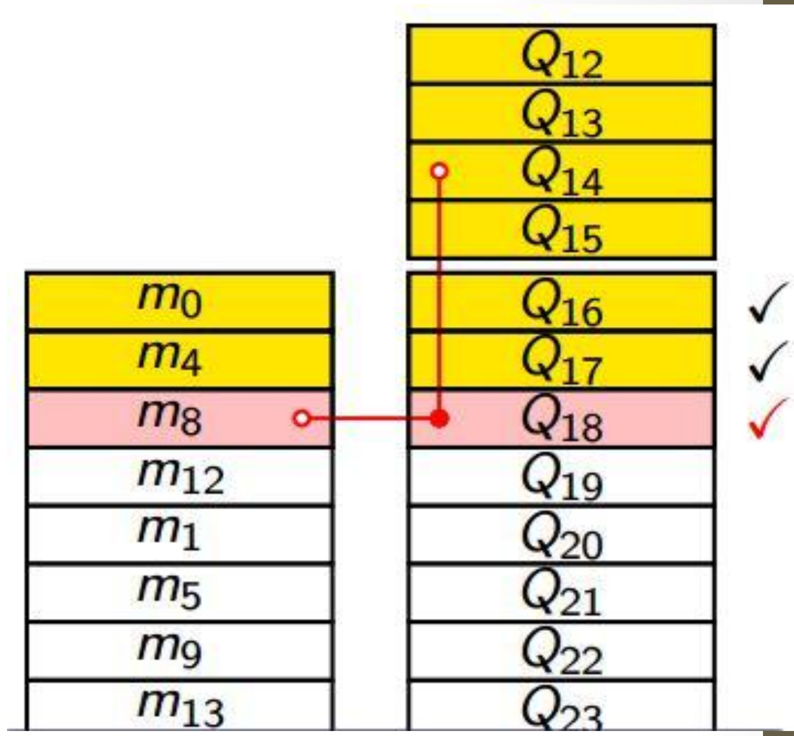
Approach to satisfy condition in the second round



- Choose the end of the first round.
- Choose m_i to satisfy both the conditions.
- **Fill the first round**.

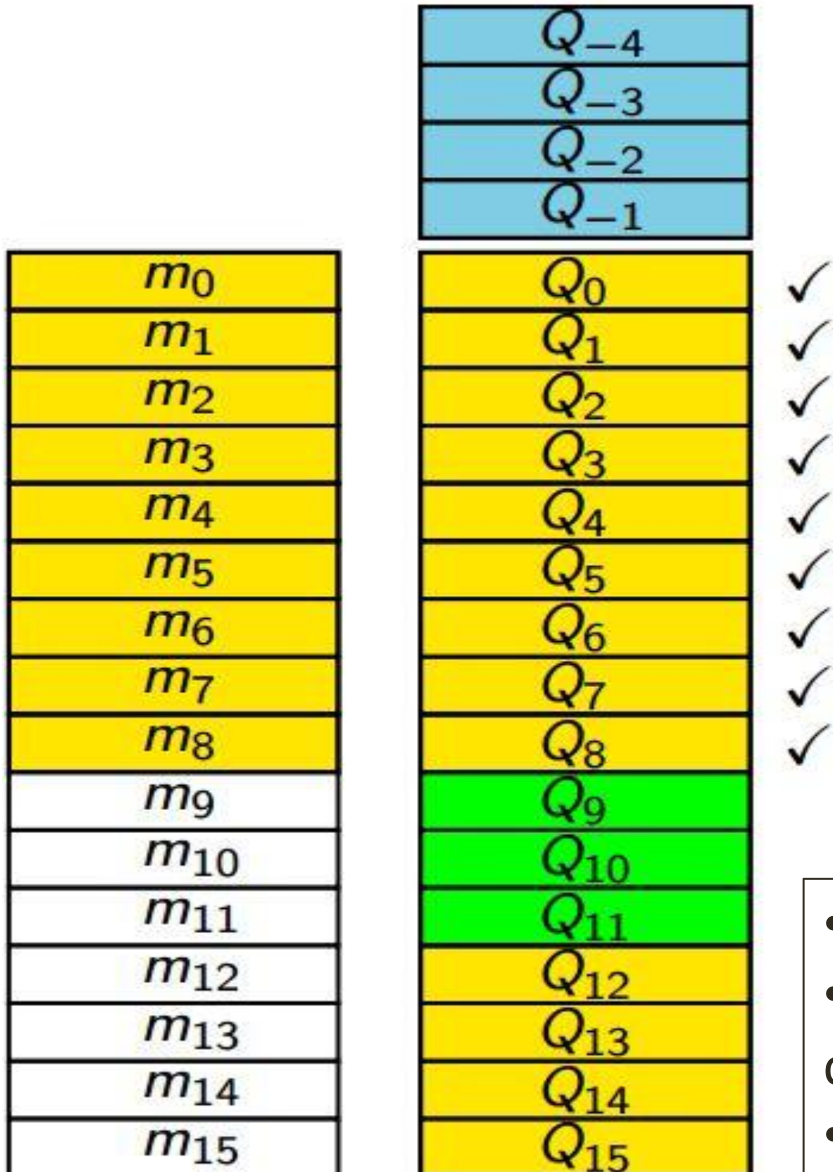
Approach to satisfy condition in the second round

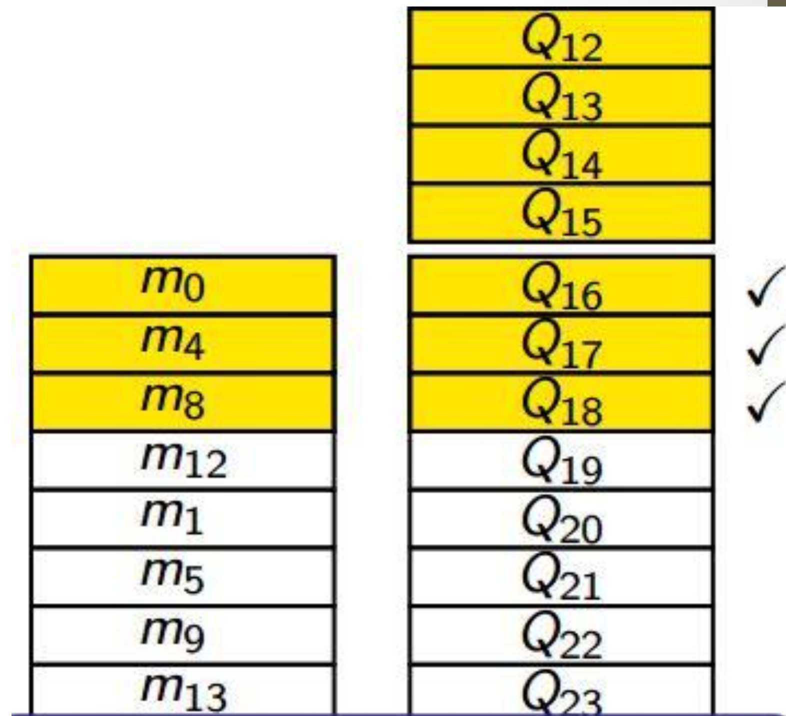




- Choose the end of the first round.
- **Choose m_i to satisfy both the conditions.**
- Fill the first round.

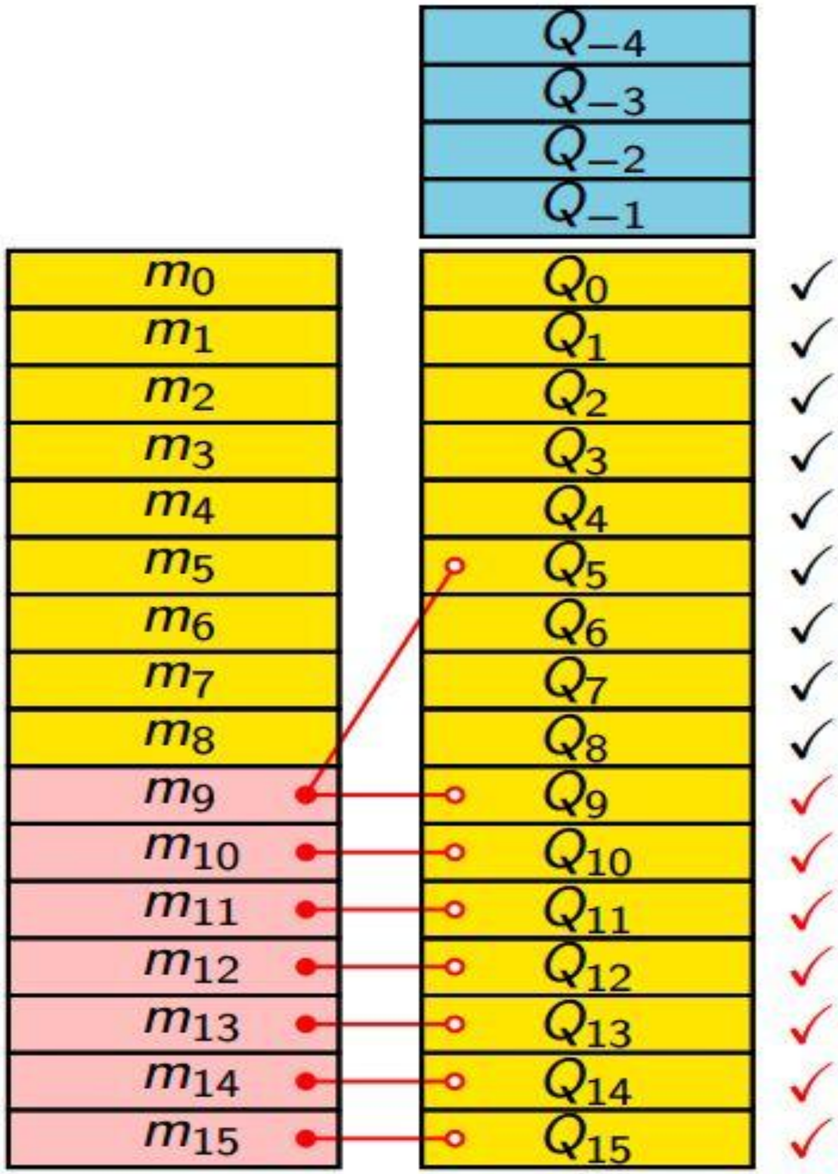
Approach to satisfy condition in the second round

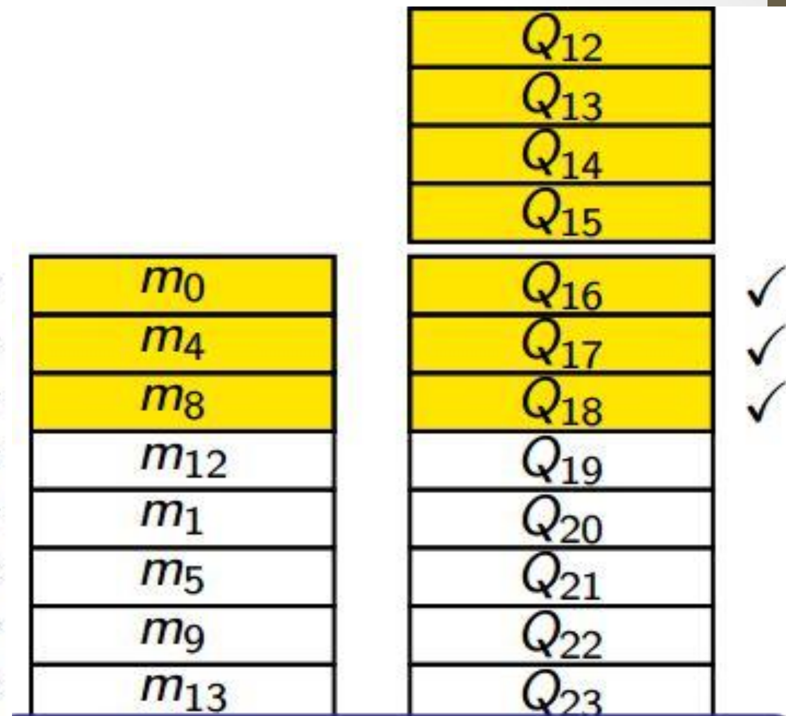




- Choose the end of the first round.
- Choose m_i to satisfy both the conditions.
- **Fill the first round**.

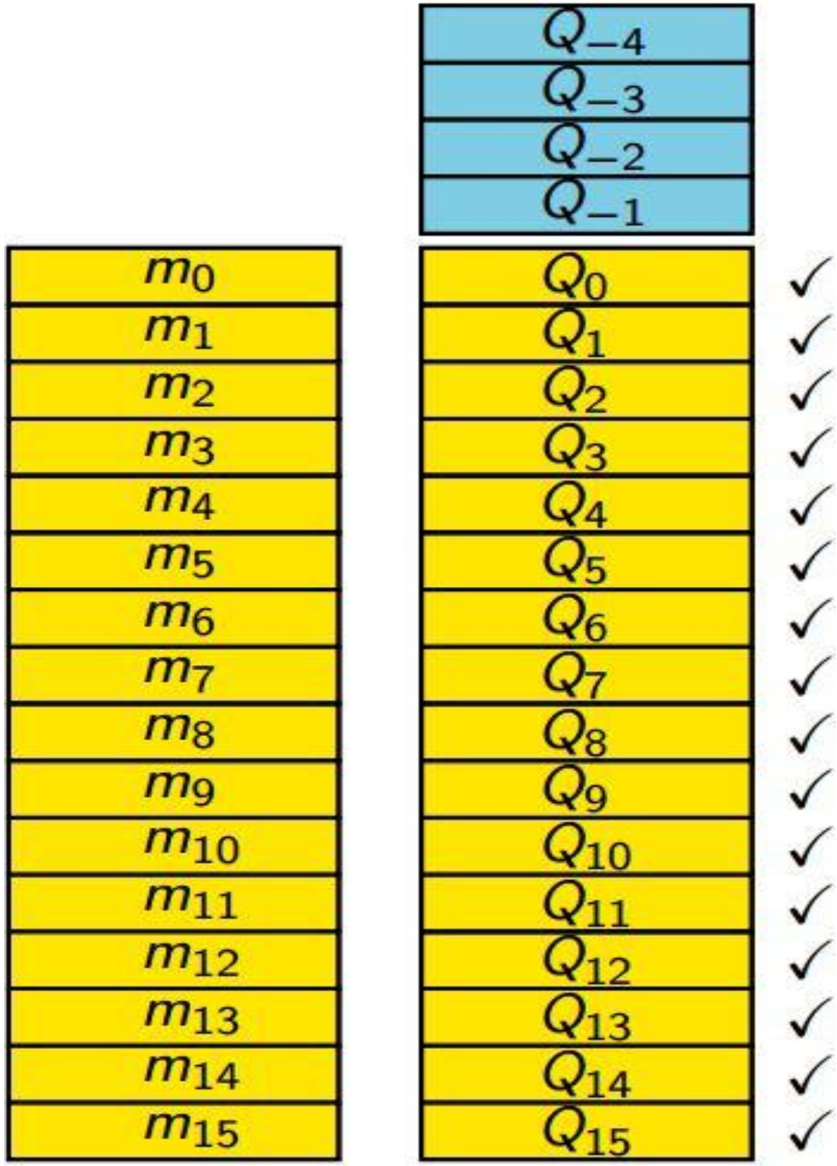
Approach to satisfy condition in the second round

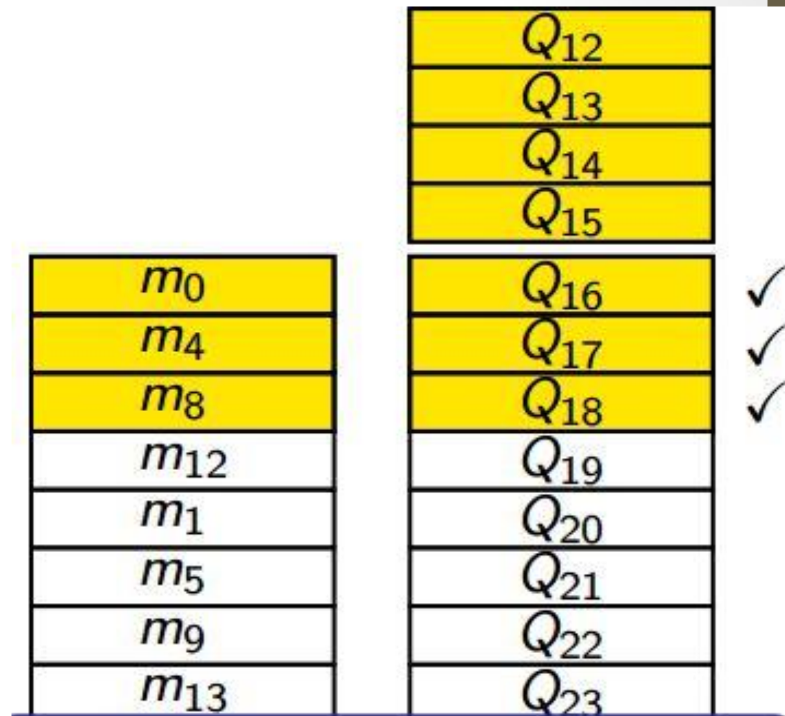




- Choose the end of the first round.
- Choose m_i to satisfy both the conditions.
- **Fill the first round**.

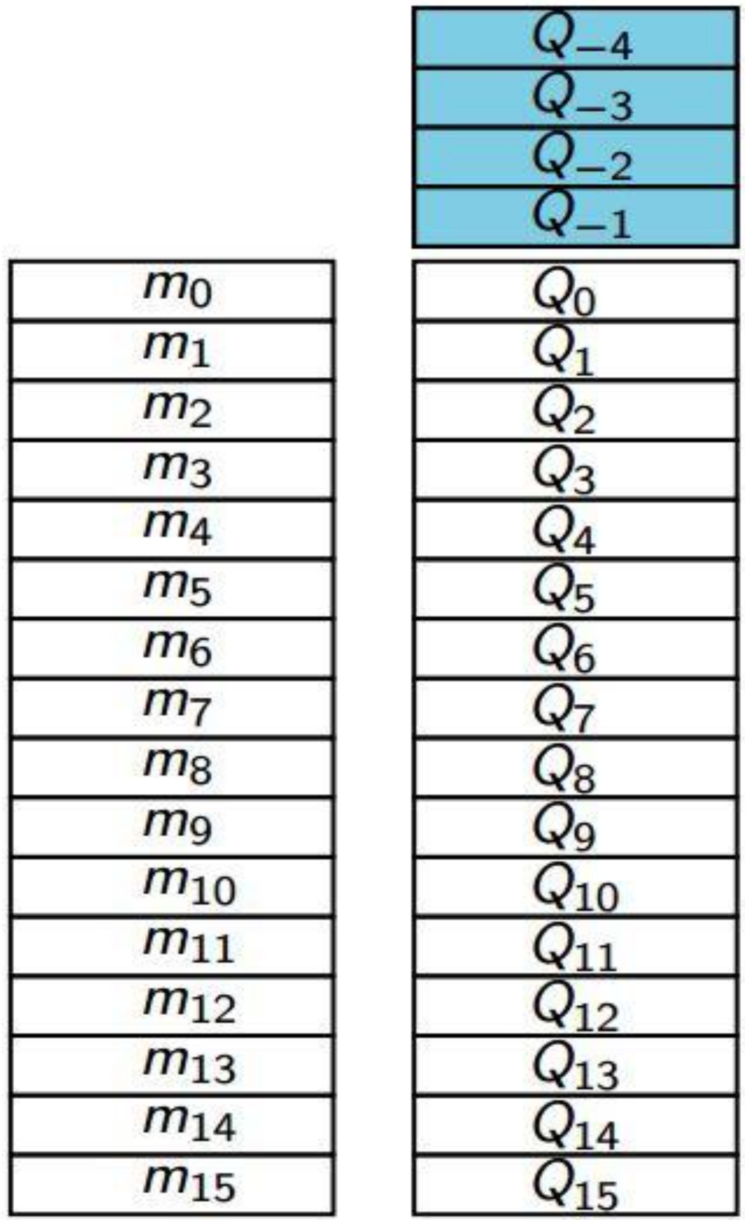
Approach to satisfy condition in the second round

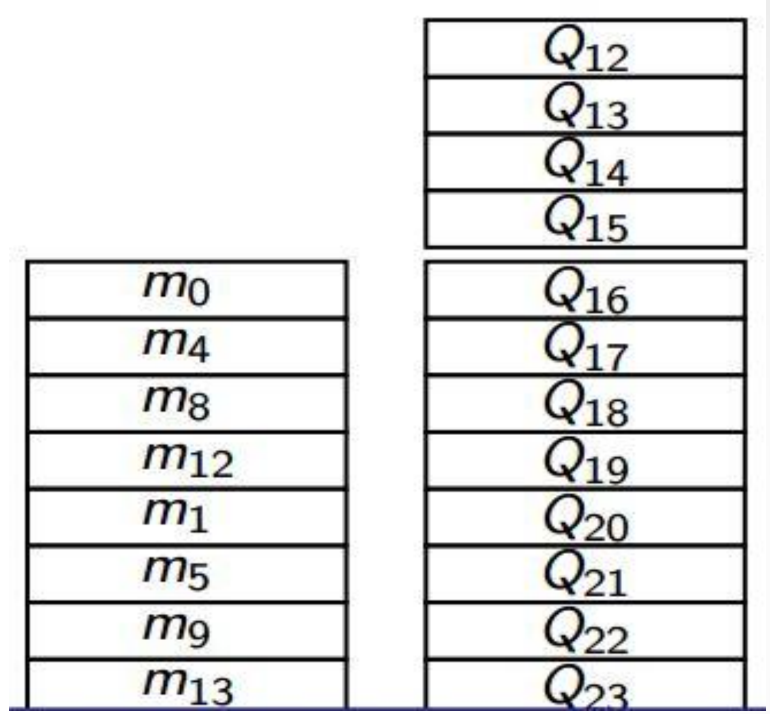




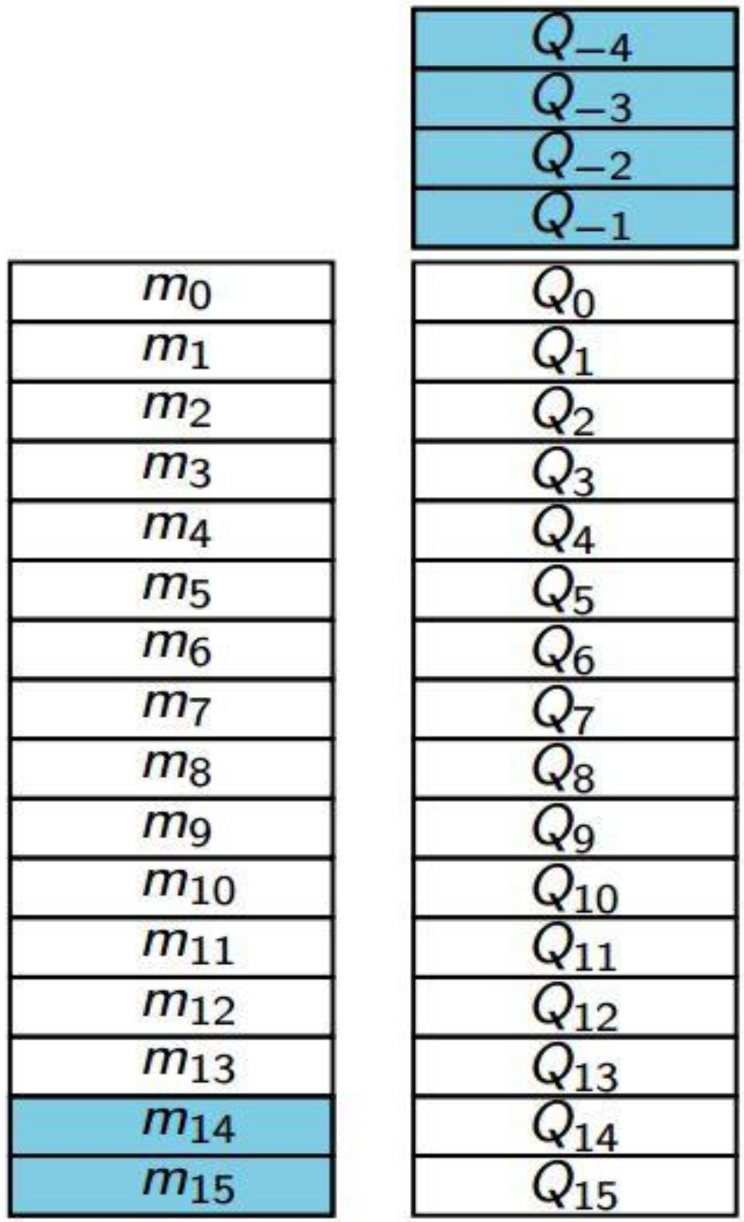
- Choose the end of the first round.
- Choose m_i to satisfy both the conditions.
- Fill the first round.

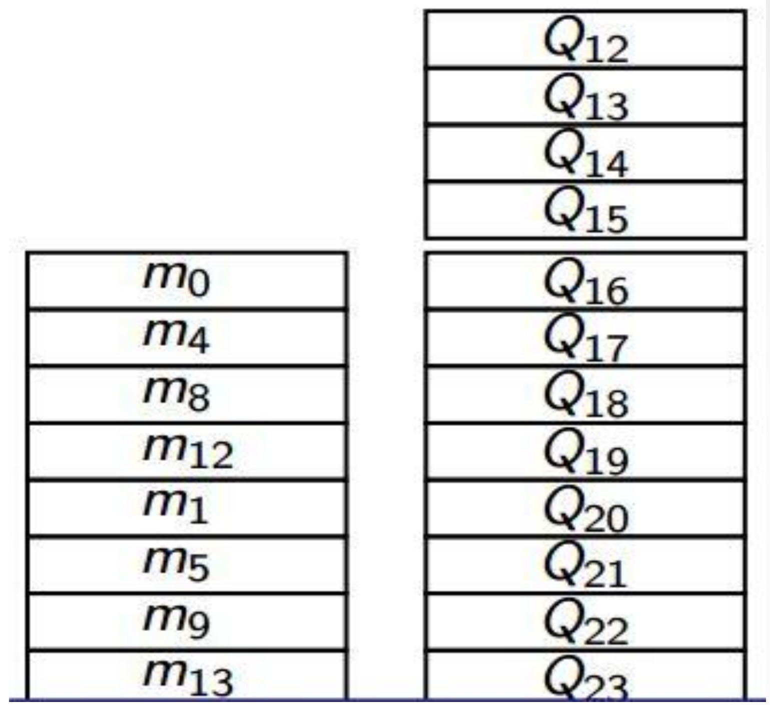
Choosing a part of the message





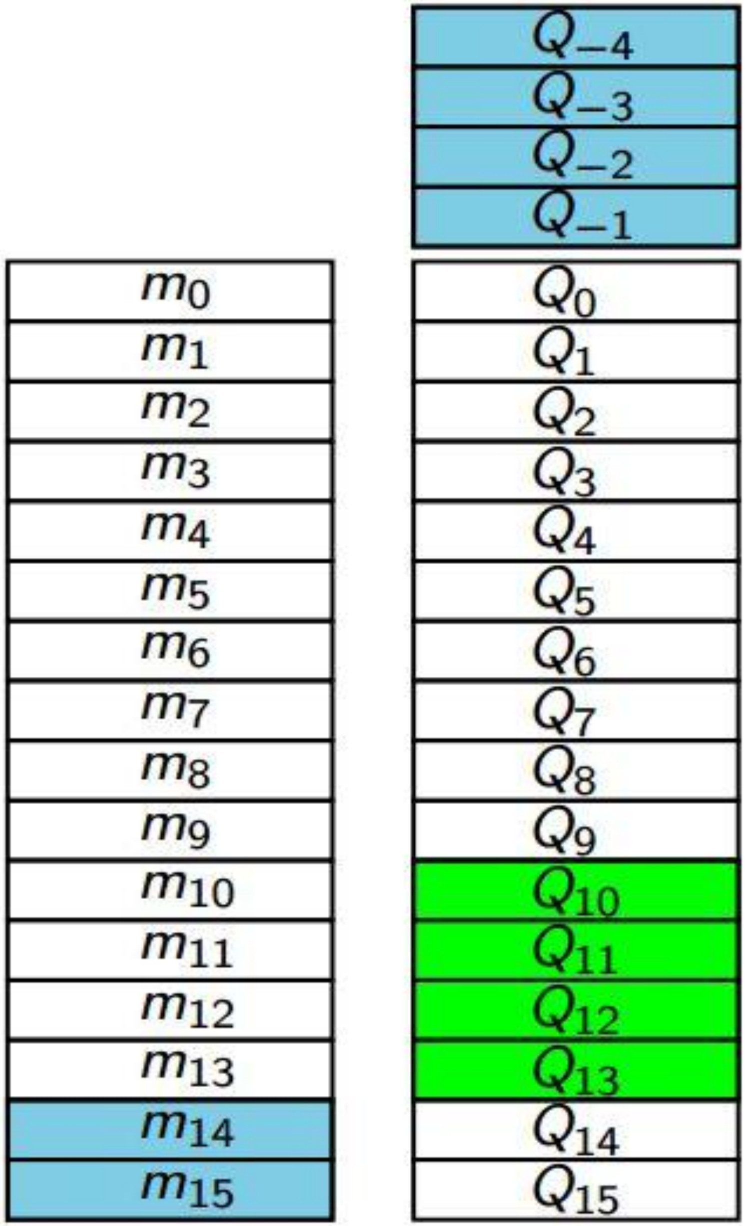
Choosing a part of the message

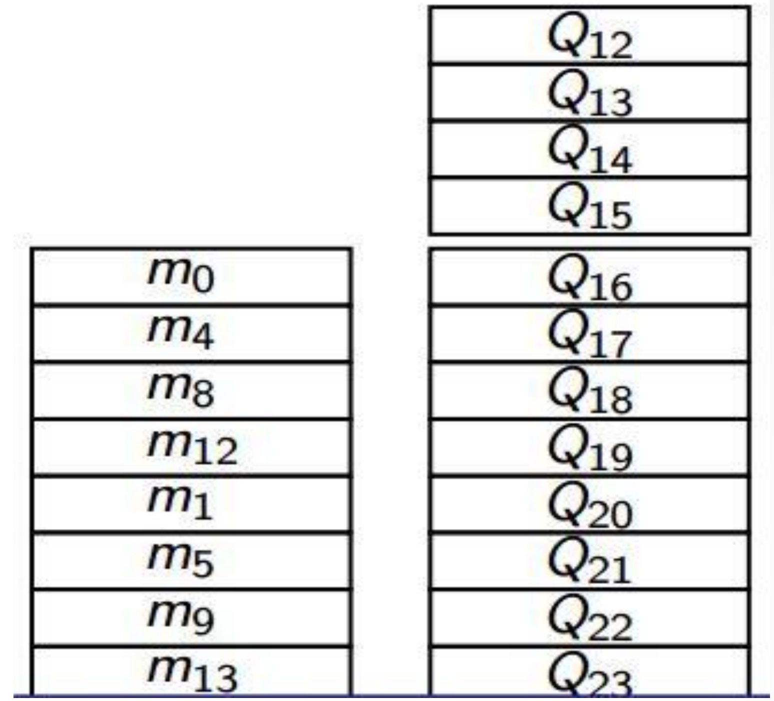




1) Choose the end of the message to be fixed.
Here **t=2.**

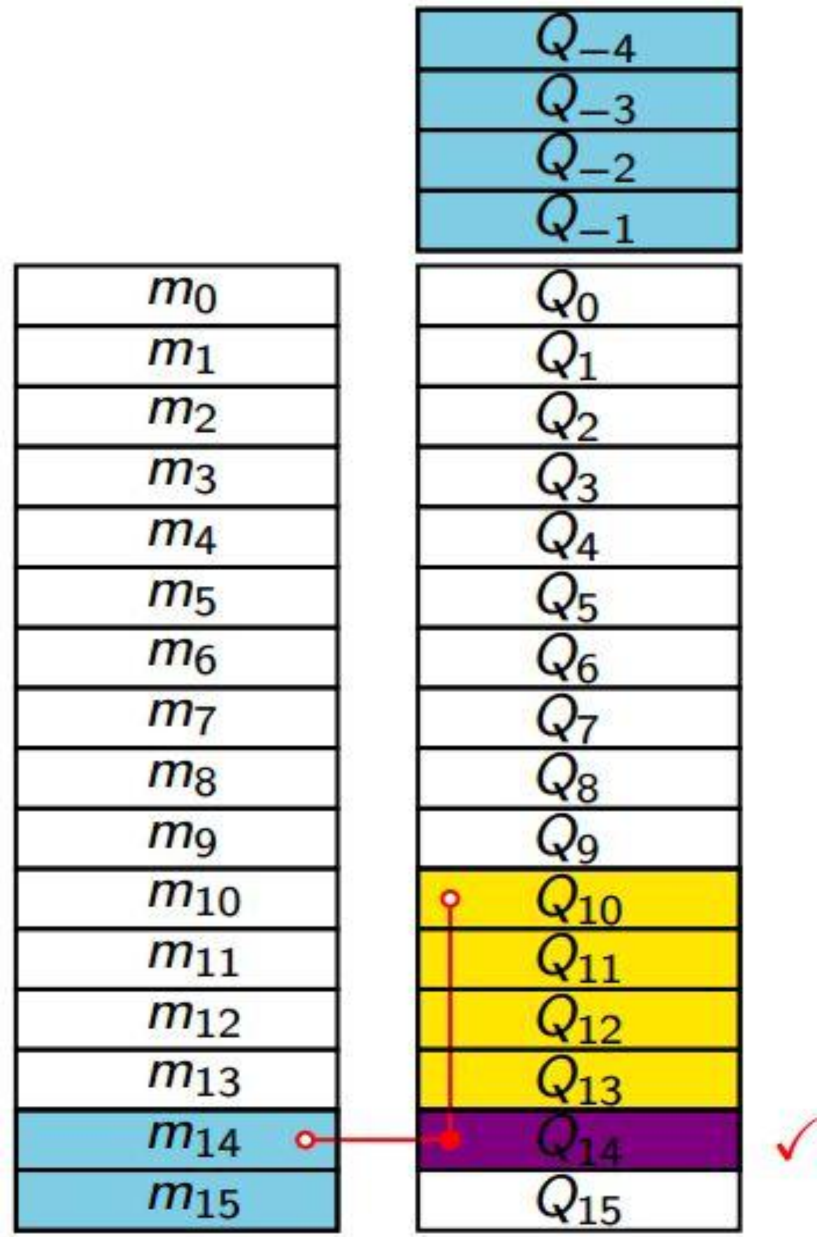
Choosing a part of the message

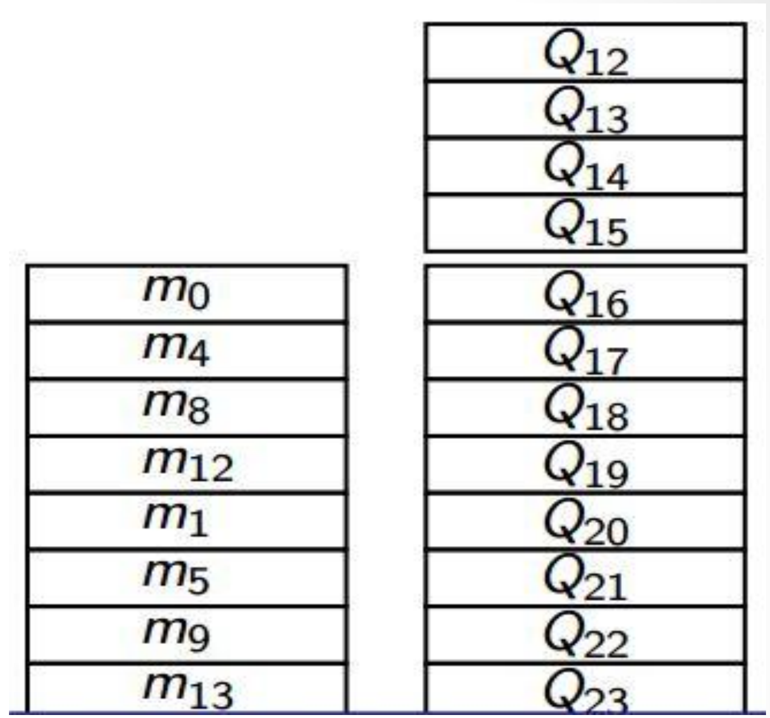




2) Choose the values Q_{12-t}, Q_{13-t}, Q_{14-t} , Q_{15-t} such that it satisfies the conditions.
Here we choose the values Q_{10}, Q_{11}, Q_{12}, Q_{13}.

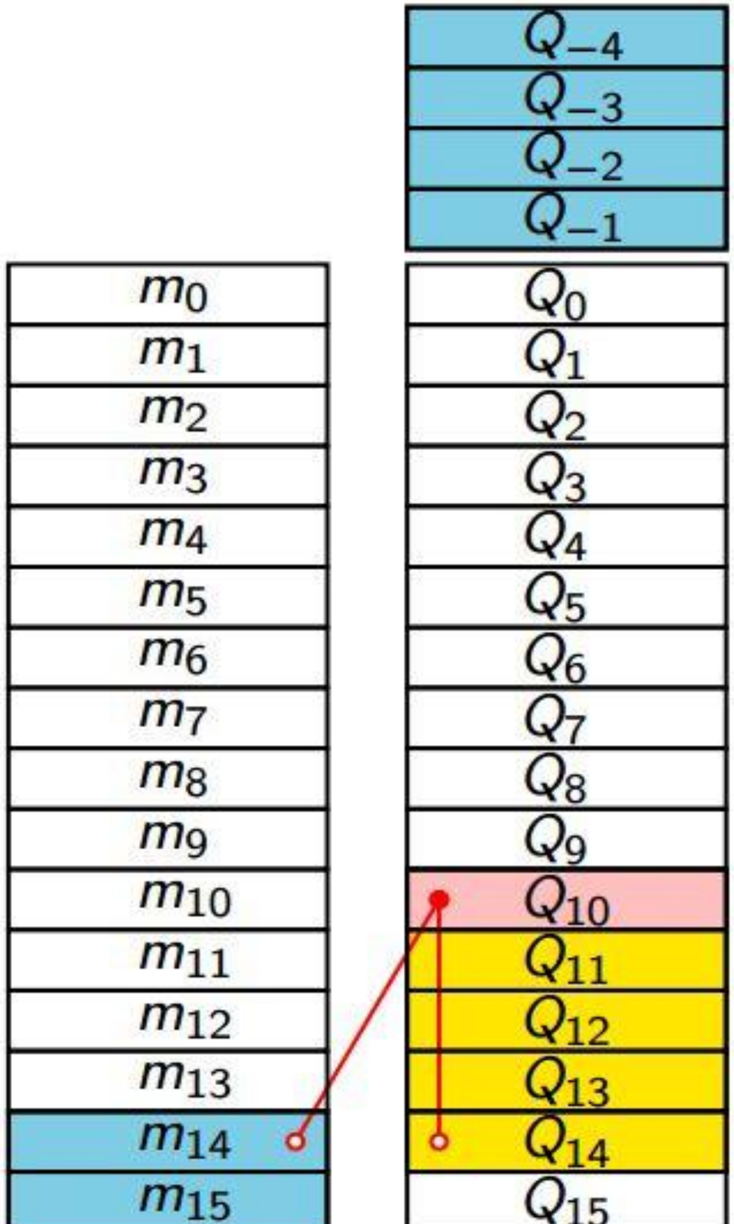
Choosing a part of the message

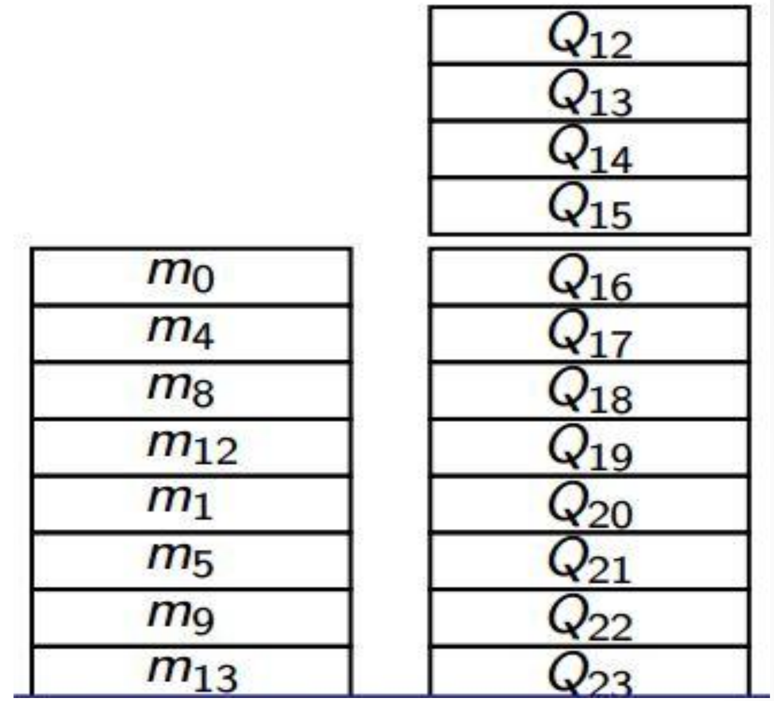




3) Compute Q_{16-t}.
Here we compute Q_{14}.

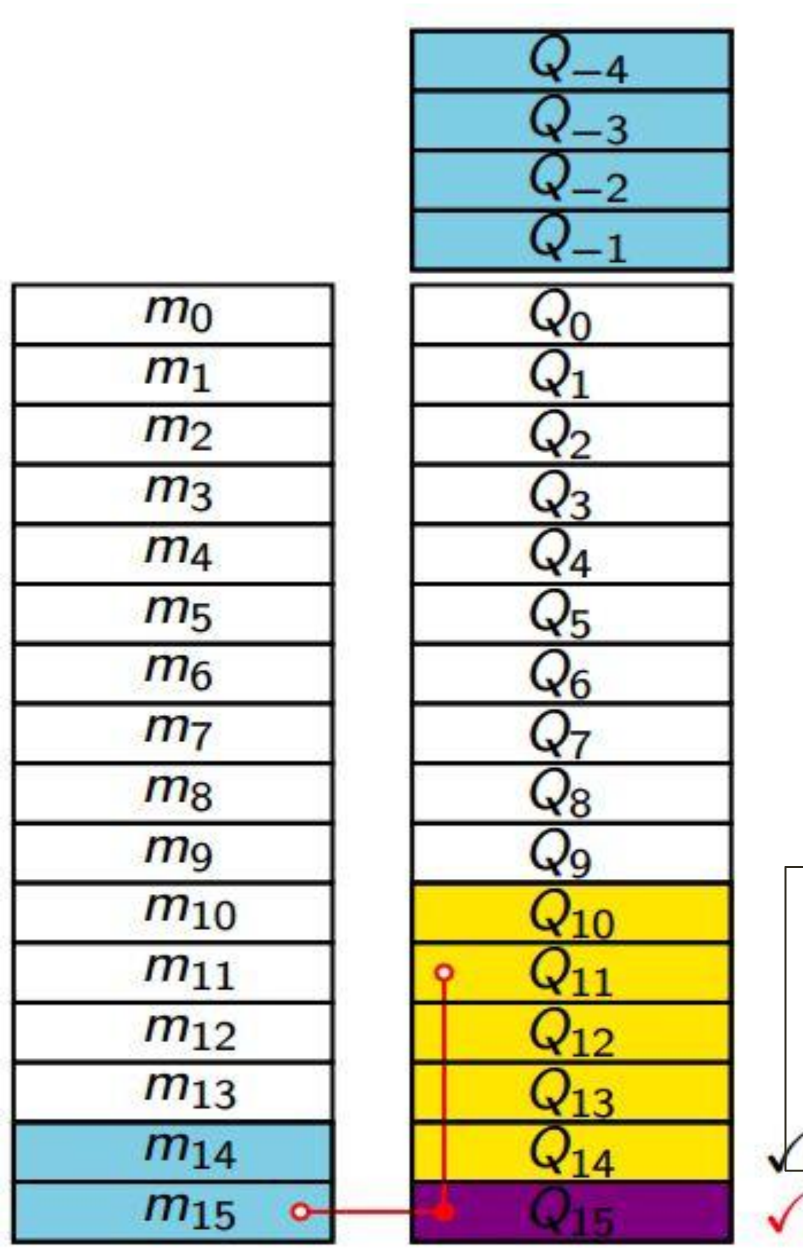
Choosing a part of the message

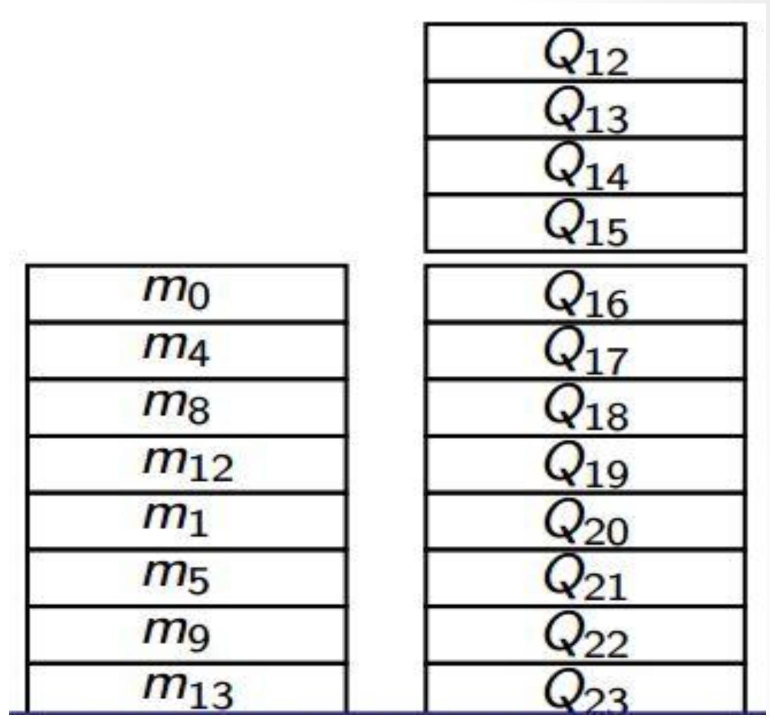




4) Re-compute $Q_{12\text{-}t}$.
Check if condition on $Q_{12\text{-}t}$ holds or not. If it does not hold, choose another set of values for $Q_{12\text{-}t}$, $Q_{13\text{-}t}$, $Q_{14\text{-}t}$, $Q_{15\text{-}t}$.

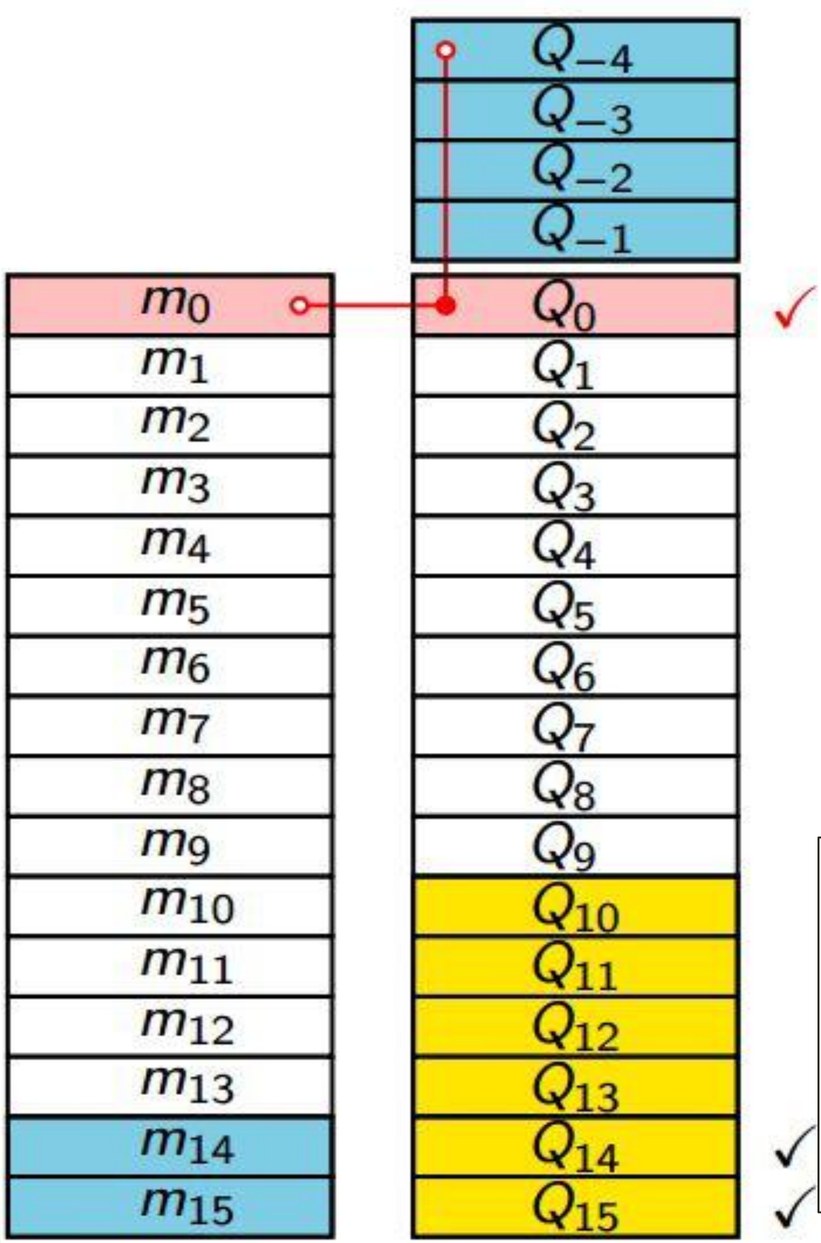
Choosing a part of the message

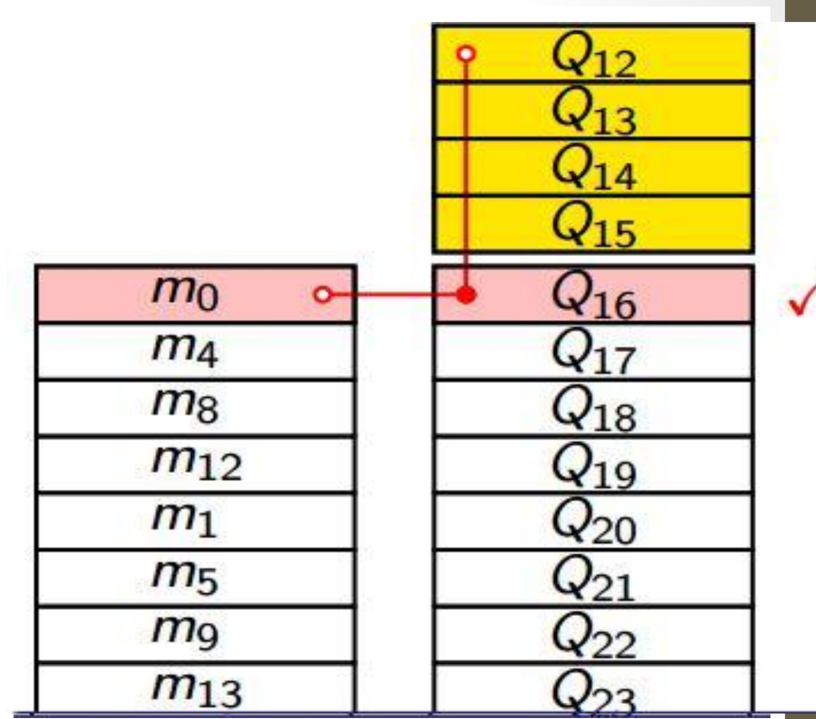




5) Compute the values Q_{17-t} to Q_{16} Check conditions on them. If it does not hold, go to step 2.
Here we compute Q_{15}.

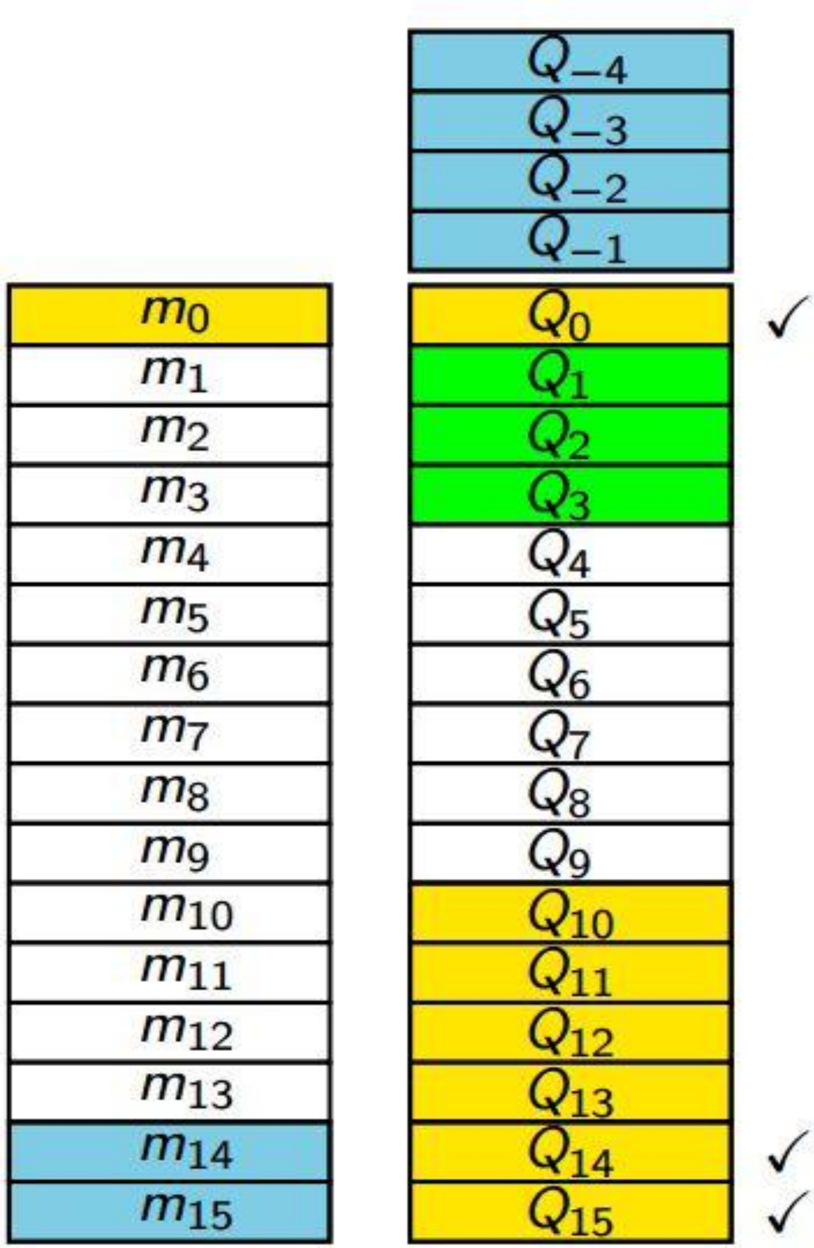
Choosing a part of the message

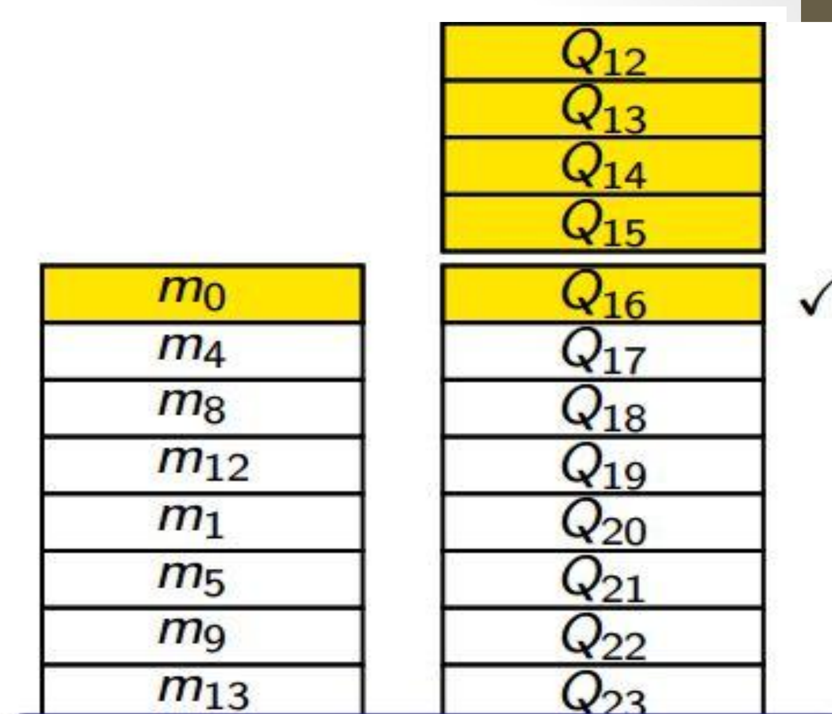




6) Compute Q_{16} from m_0.
Re-compute m_0 from Q_{16}.
Compute Q_0 from m_0.
Conditions on Q_0 and Q_{16} should hold true

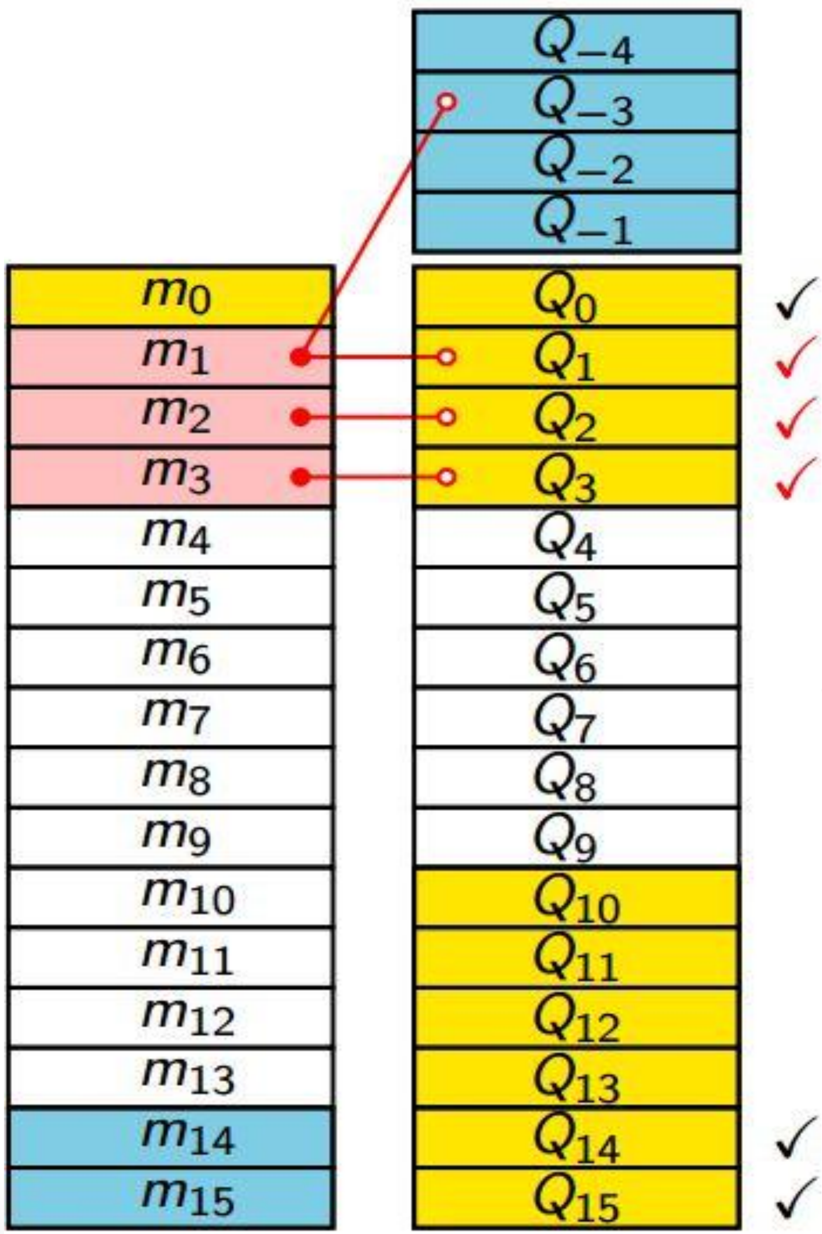
Choosing a part of the message

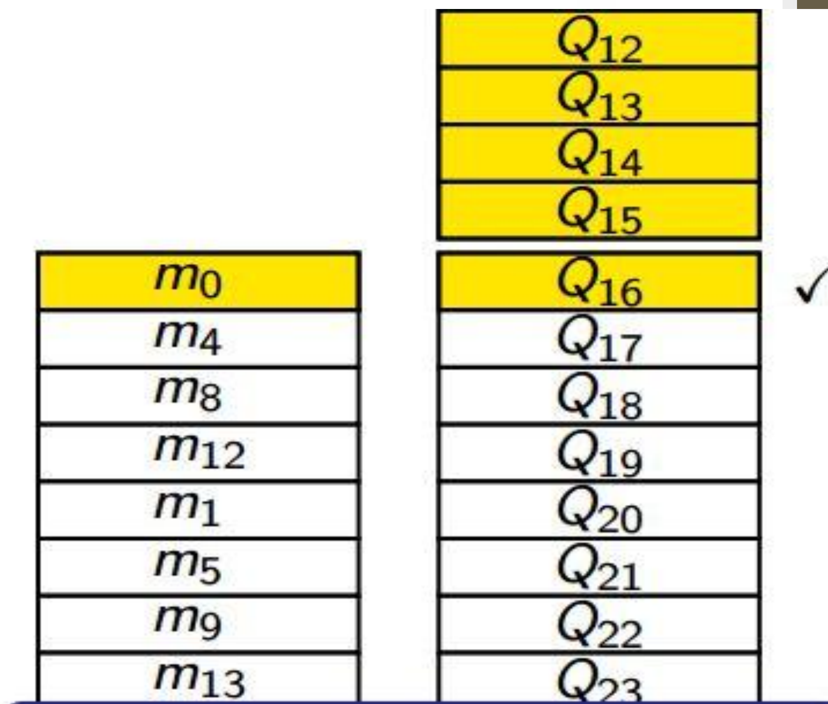




7) Fill the first round.

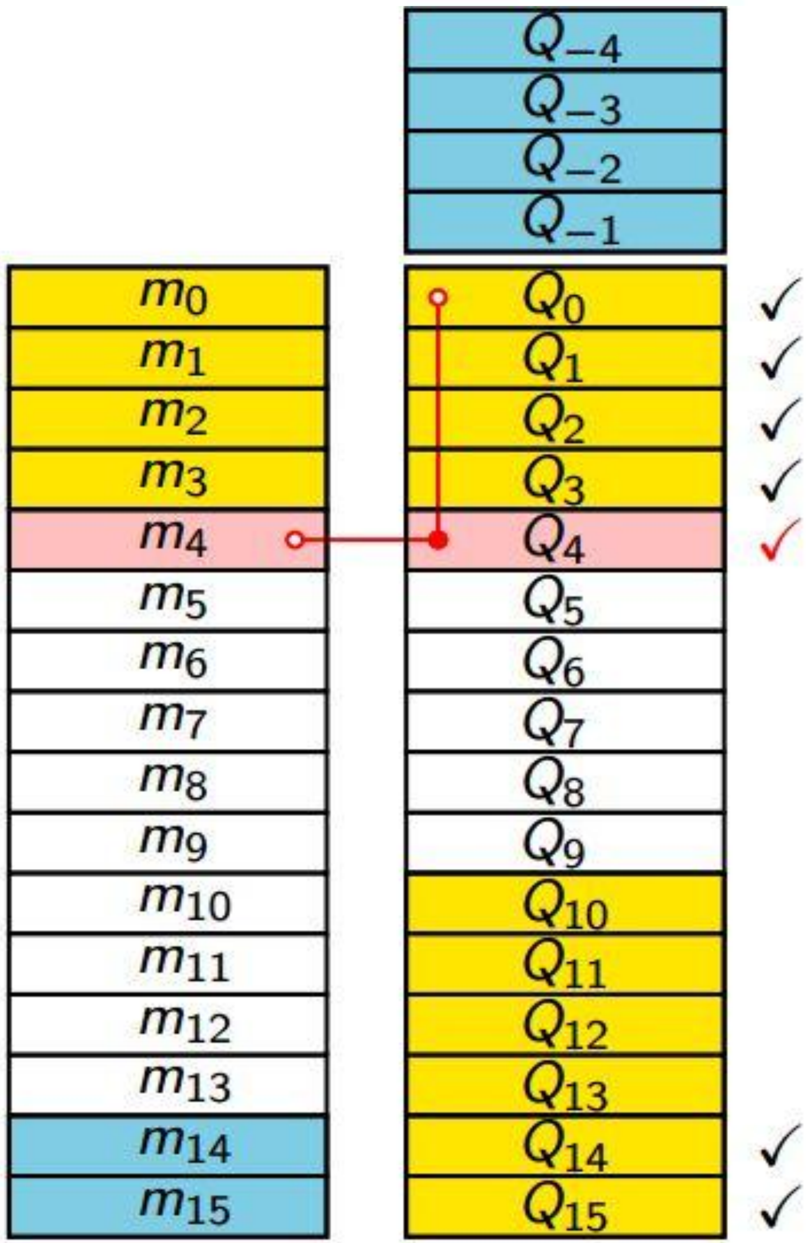
Choosing a part of the message

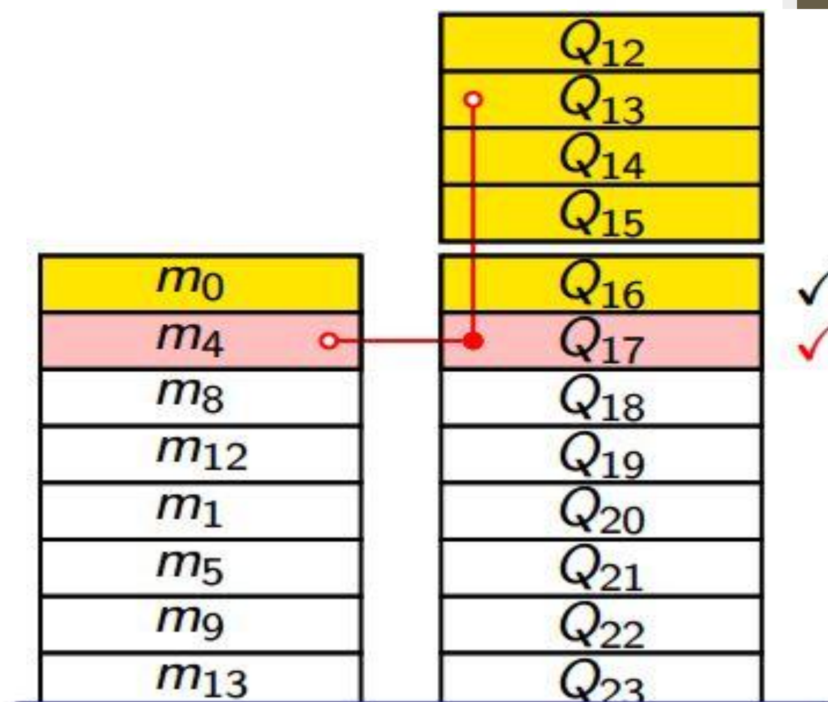




8) Fill the first round.

Choosing a part of the message



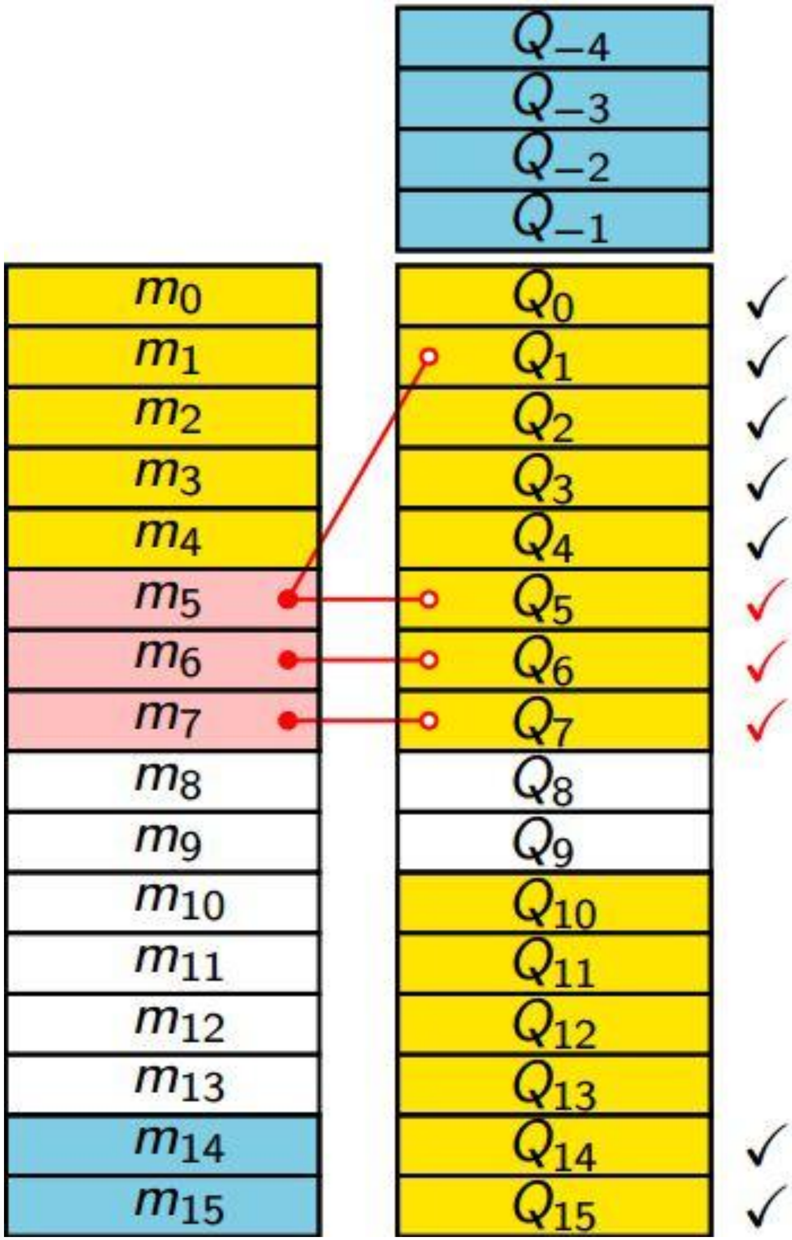


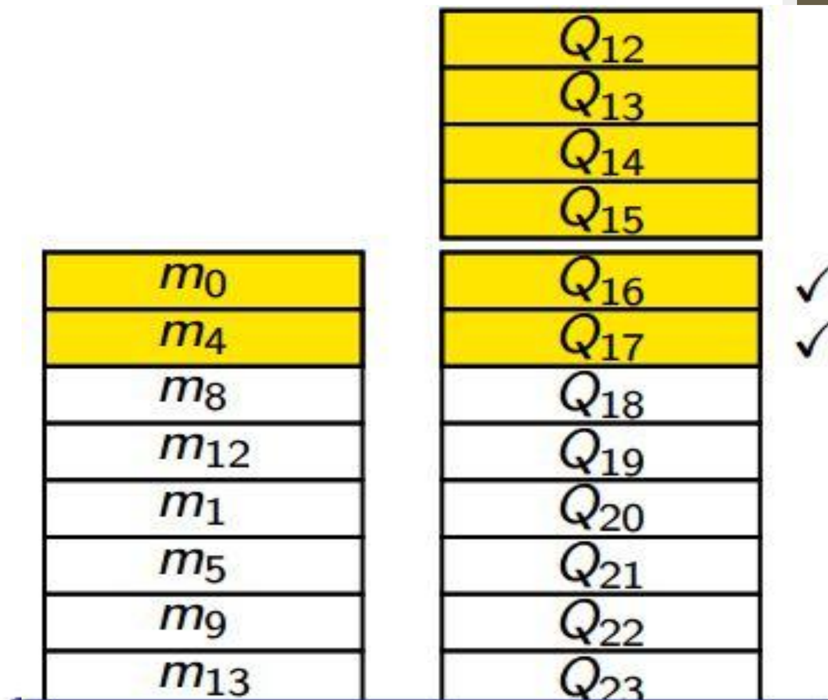
9) Compute Q_{17} from m_4.
Re-compute m_4 from Q_{17}.
Compute Q_4 from m_4.

Choosing a part of the message

10) Fill the first round.

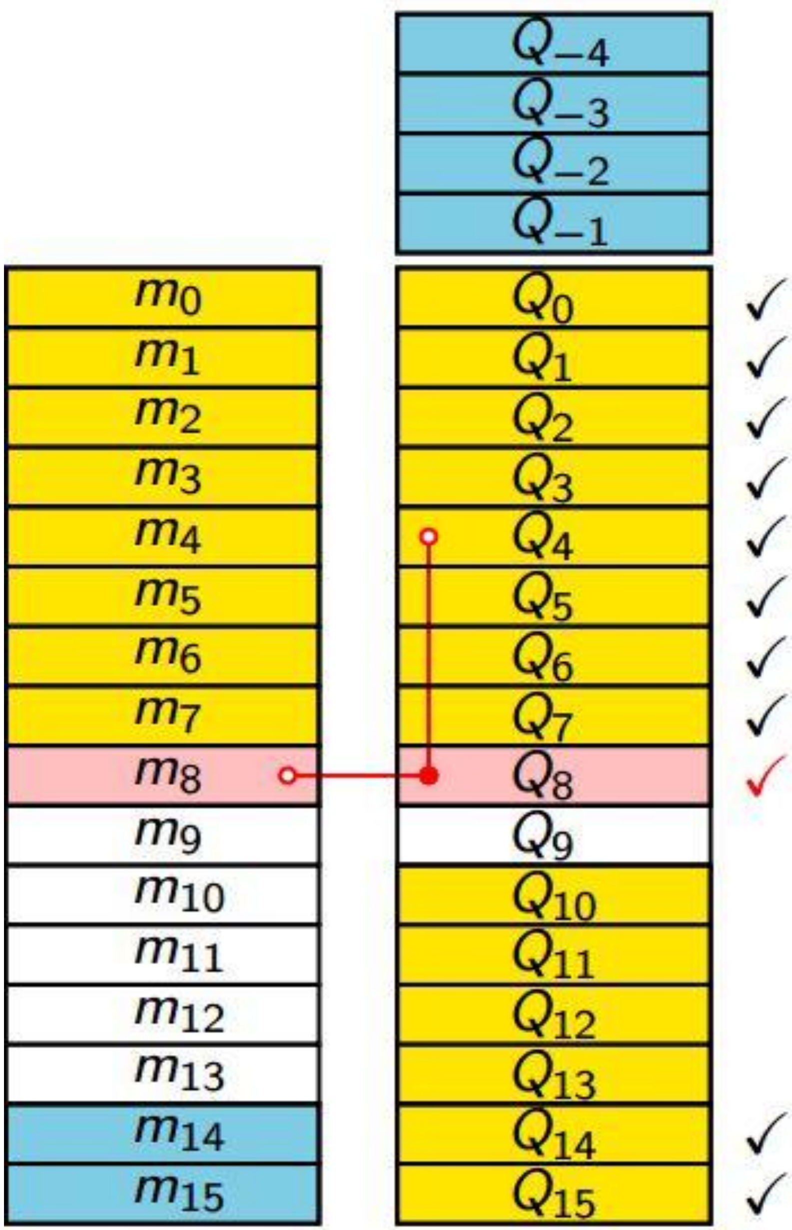
Choosing a part of the message

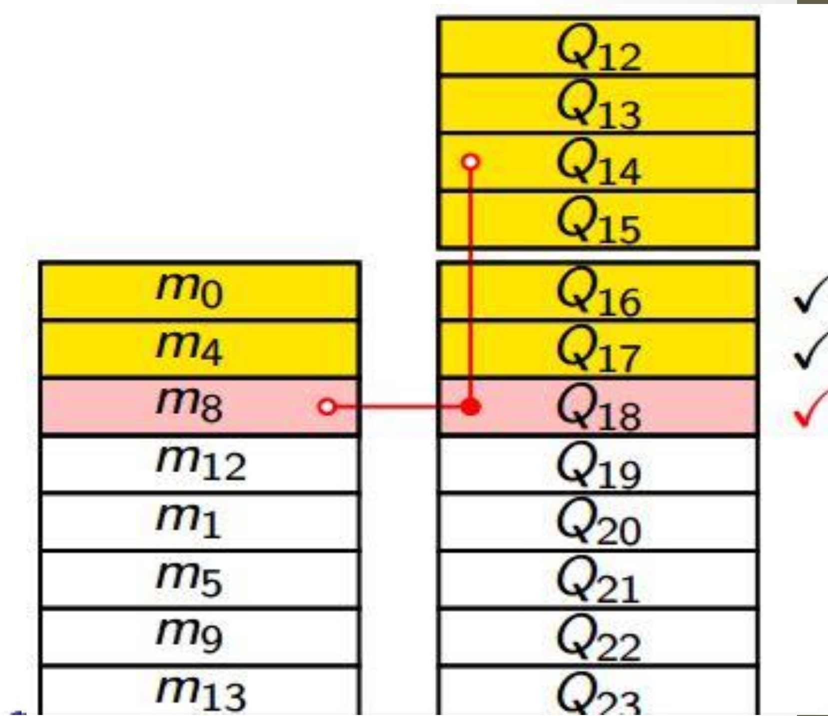




11) Fill the first round.

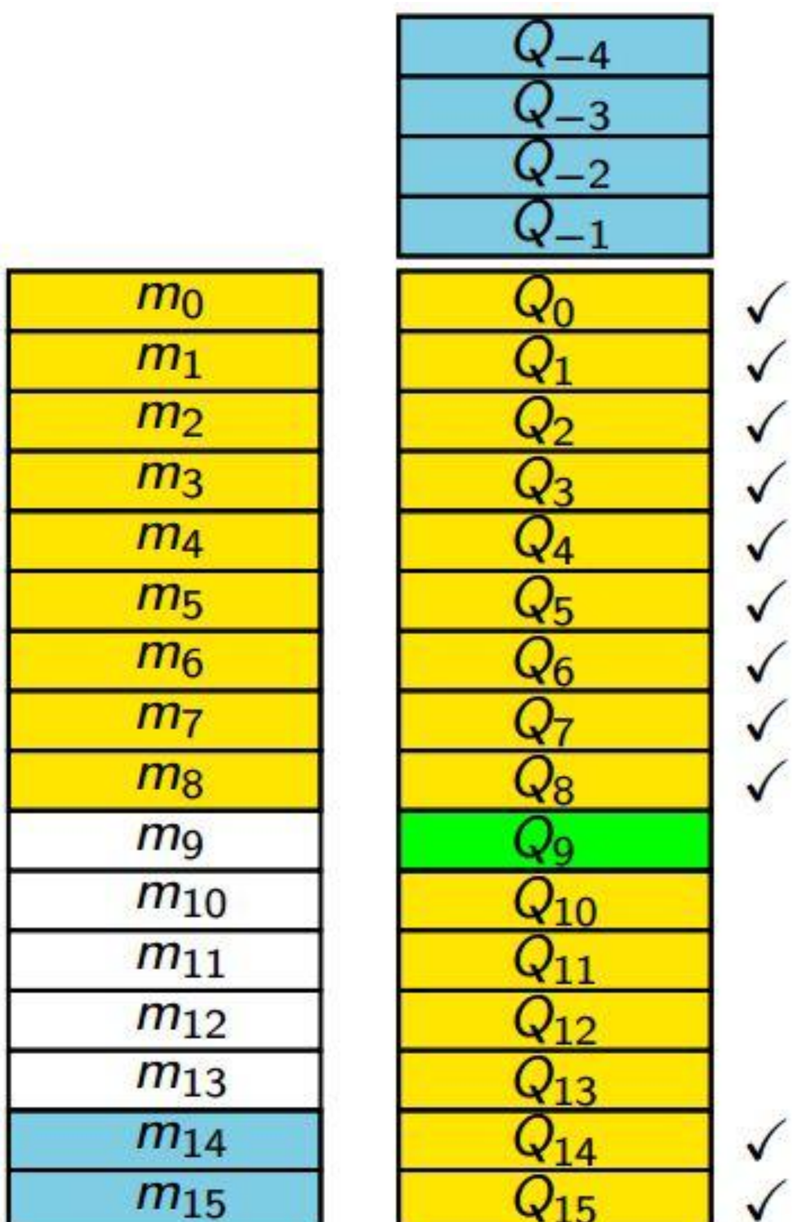
Choosing a part of the message

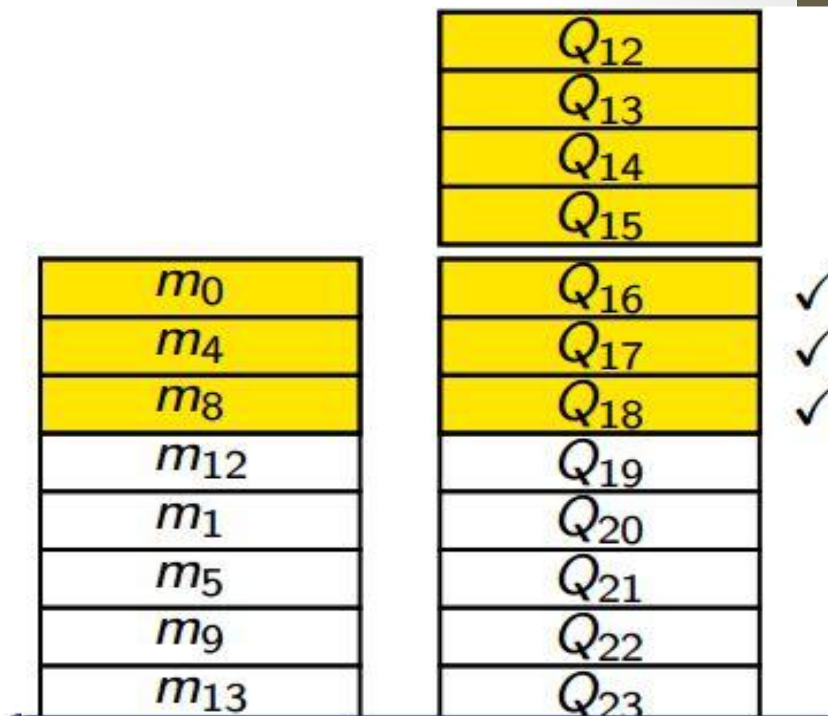




12) Compute Q_{18} from m_8.
Re-compute m_8 from Q_{18}.
Compute Q_8 from m_8.

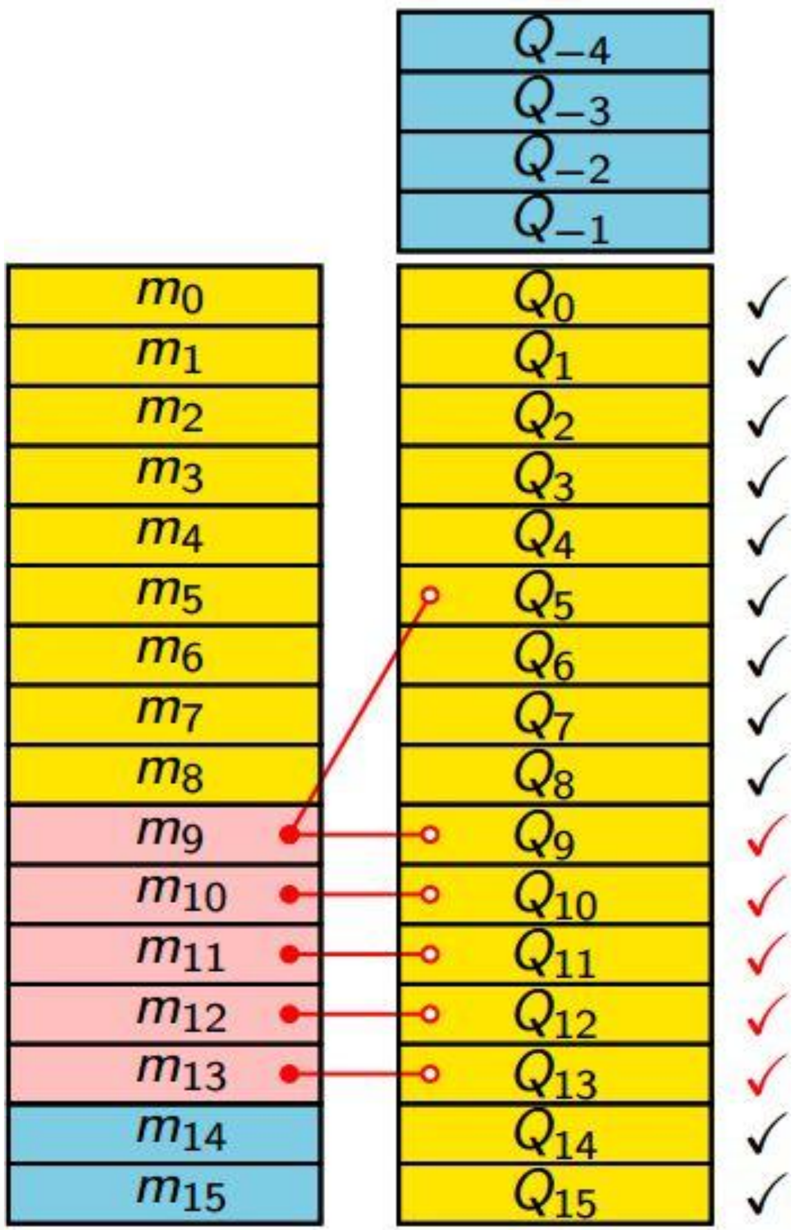
Choosing a part of the message

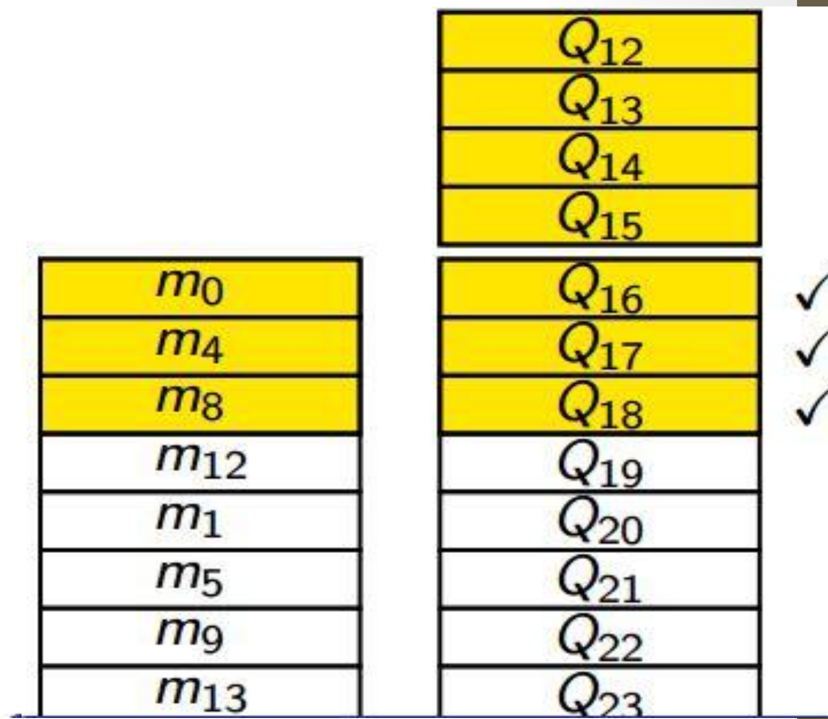




13) Fill the first round.

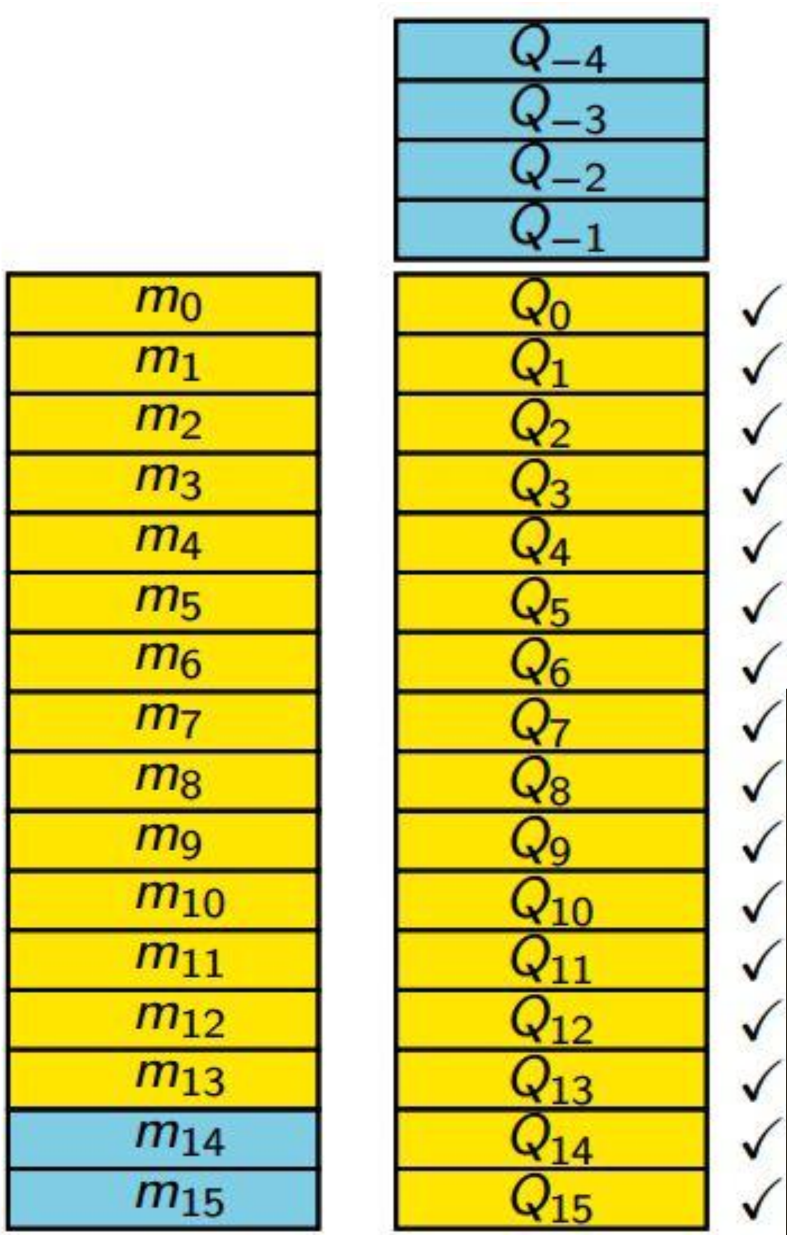
Choosing a part of the message

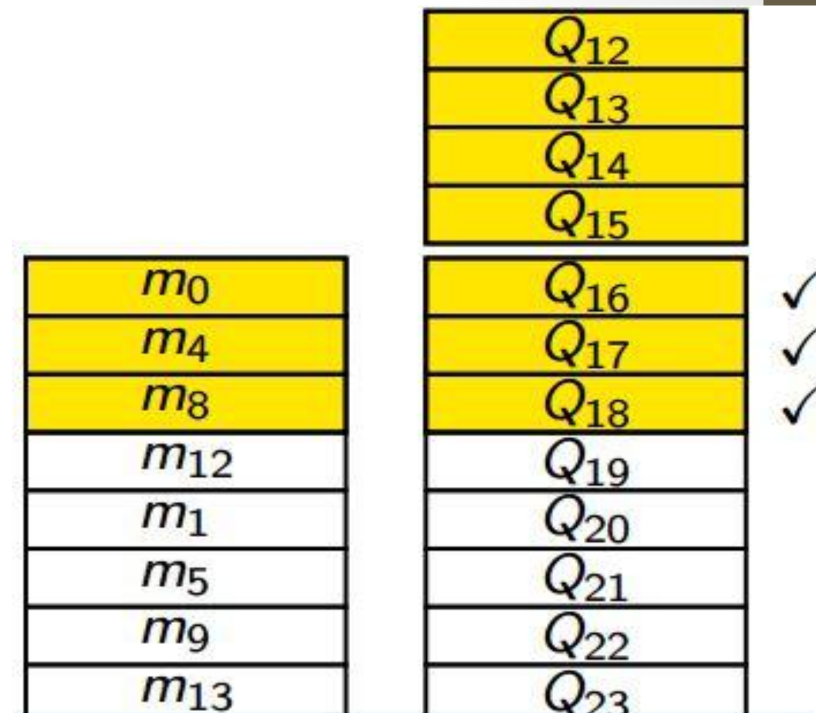




14) Fill the first round.

Choosing a part of the message





15) Compute Q_i from p_c (19) and to p_v (24) using the previous Q_i's. If the condition on Q_i does not satisfy then go to step 13.

16) Using tunnels, find a message which satisfies all Q_i's upto the end of the round.

Message Freedom

- Using this approach, one can choose:-
  - last three message words in a one-block MD4 collision
  - three specific message words on a two block MD5 collision.
- Collision Search
  - First block computed only once: include ‘<‘ and ‘@’ .
  - For the second block:-
    - Avoid 4 characters (>, p_0, p_1, p_2)
- We can recover 3 characters of the password.

Why only 3 words can be recovered in MD5?

- For MD5, the Wang's path uses a message difference in m_{14}.
- In order to learn i^{th} password character, we need to generate a collision where we fix the last i+1 characters (three characters of the password and '>')
- Due to the message difference we cannot modify the message m_{14} and hence can only recover 3 characters of a password.

Attack Complexity

- Assume, password is 8 char. long and each char. has 6 bits of entropy.
- Generate **$3*2^5$ collisions** and wait for about **$3*2^6$ identifications.**
- If each collision takes 5 sec. to generate, then attack will take about 3 hours.

Note : This is not clearly understood .

APOP Attack in practice

- More than 10% use POP ,out of which about 4% use APOP (not a negligible number)
- Some mail user agents give the freedom to select the authentication method to the server – Attacker can claim to support only APOP
- Colliding messages cannot be found for ASCII – but most of the mail clients are non RFC-compliant and only check for only condition 1 & 3 on slide 5

Clients	Status
Netscape/ Thunder bird / Mozilla	Attack works
Qualcomm Eudora	Attack works
Mutt	Attack works
Novell Evolution	Attack works
Fetchmail	Attack works
Kmail	**Attack Fails**
Microsoft Exchange/Outlook / Outlook express	No APOP support
Apple Mail	No APOP support

References

- Slides - Message Freedom in MD4 and MD5 Collisions. Application to APOP, Gaëtan Leurent
- Ch5, Applied Cryptanalysis – Breaking ciphers in real world , Mark Stamp and Richard M. Low
- Hashing in Computer Science ,Alan G.Konheim
- Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications , Klima
- Tunnels in Hash Functions: MD5 Collisions Within a Minute , Klima
- Characterizing Padding Rules of MD Hash Functions , Preserving Collision Security , Mridul Nandi
- Lecture slides of IT 325 , Winter 2012