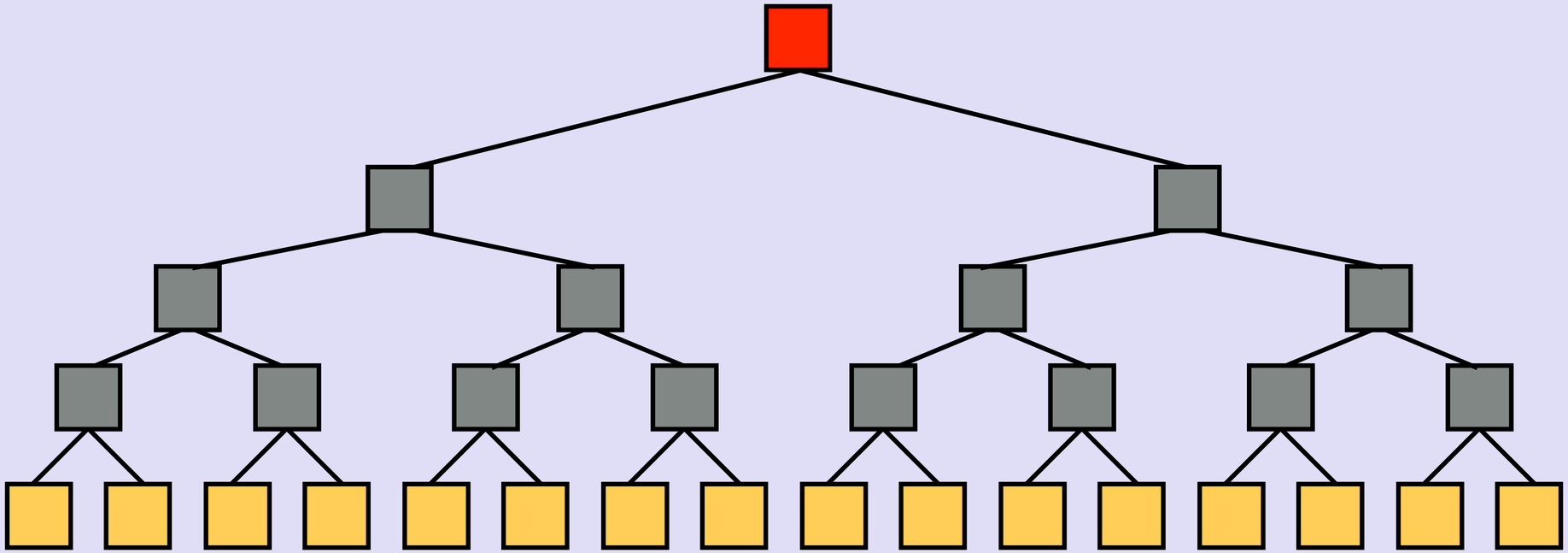
# IOTA TUTORIAL 18

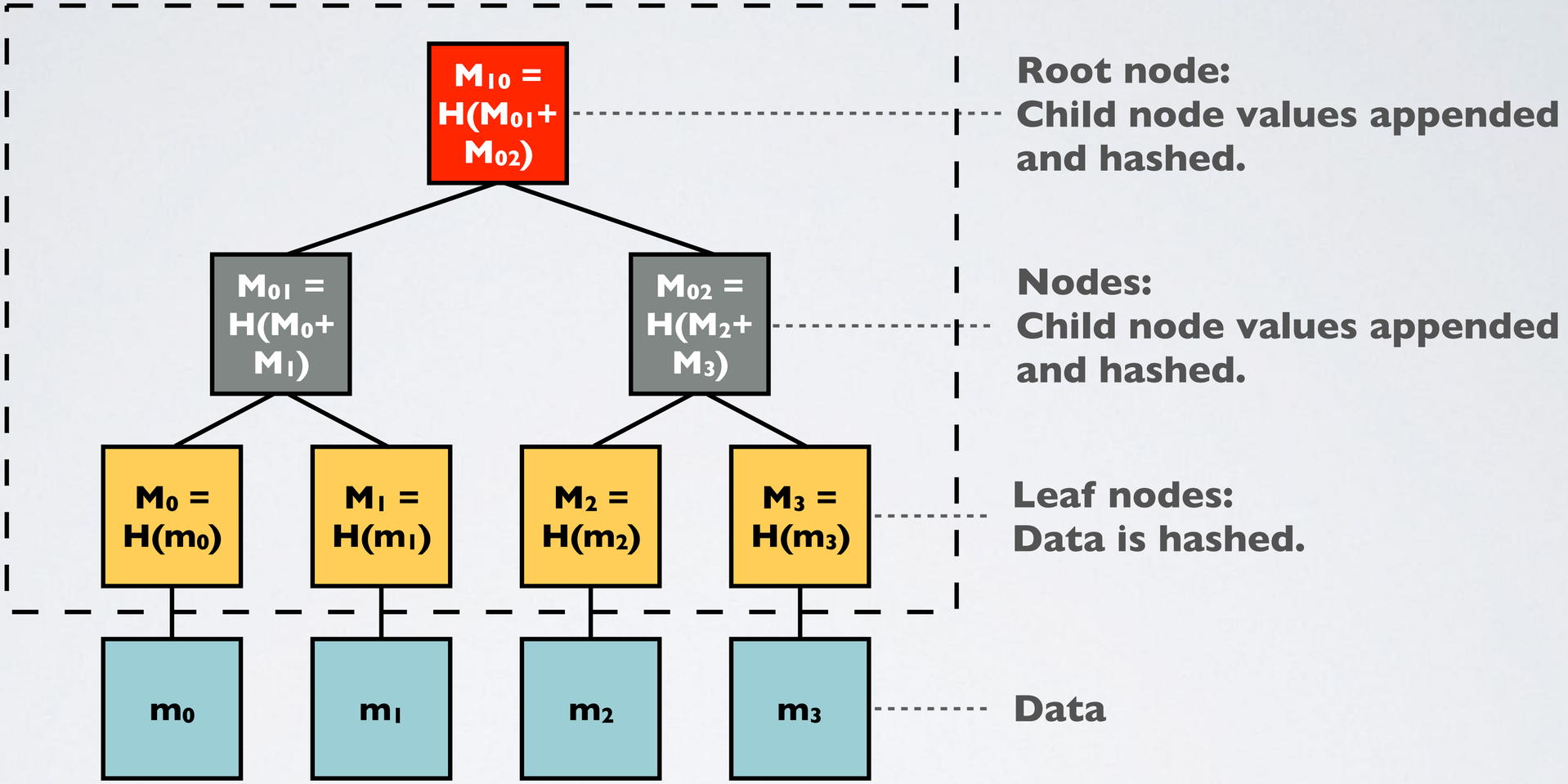
## Merkle Tree



# INTRO

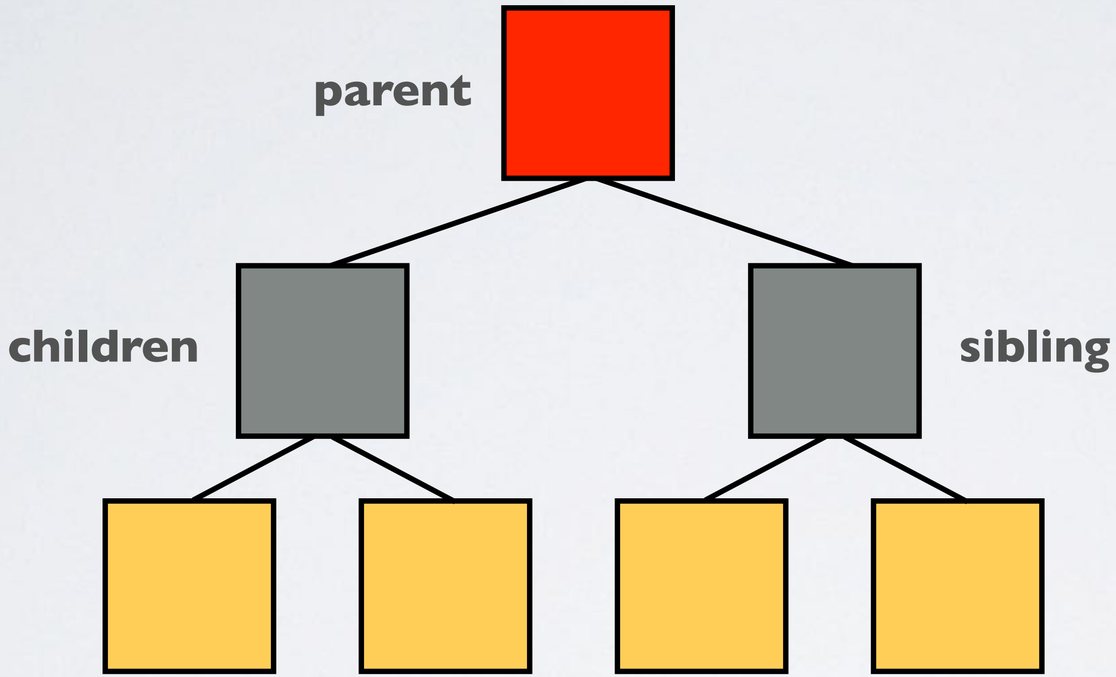
- The main objective of this video is to provide you with some basic knowledge about the Merkle tree.
- The Merkle tree is used in IOTA's Masked Authenticated Messaging.
- IOTA's Masked Authenticated Messaging will be explained in IOTA tutorial 19.

# MERKLE TREE



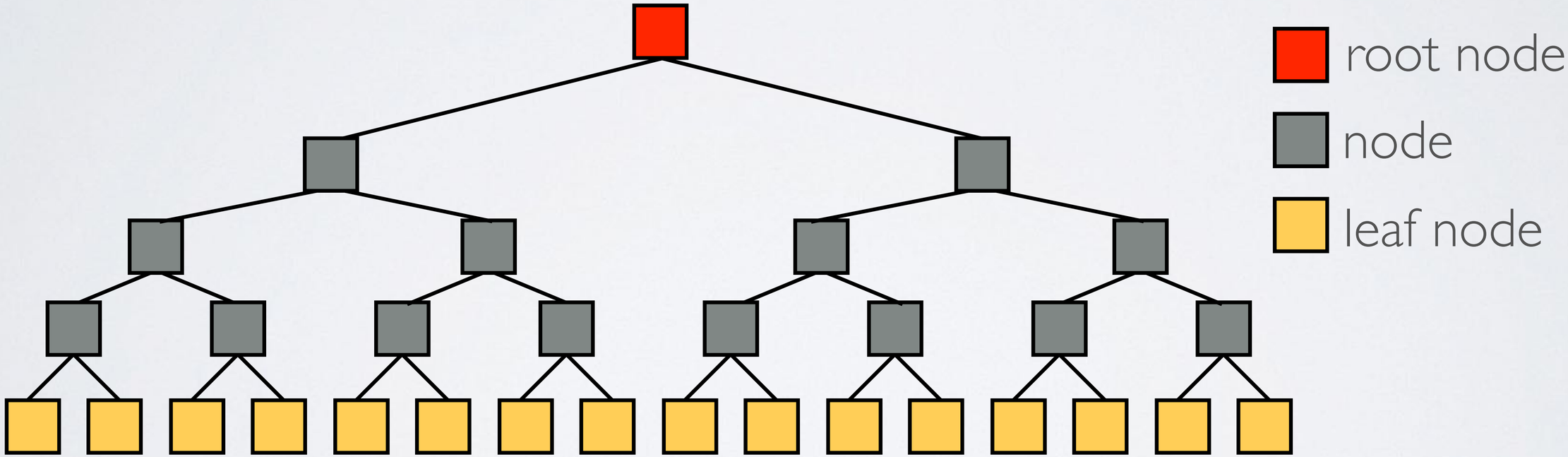

**The data (m) itself is not considered part of the Merkle tree but the HASHED data (M) is part of the Merkle tree.**

# MERKLE TREE



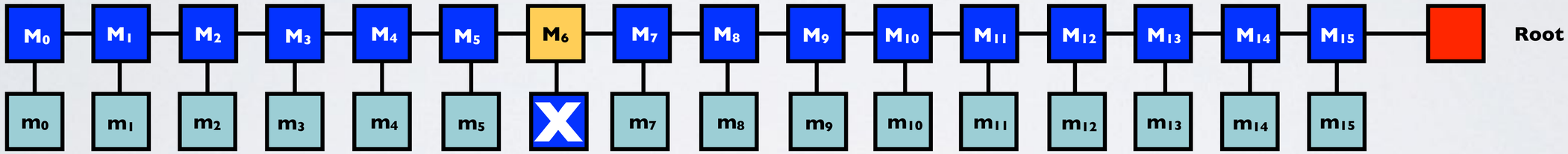
# MERKLE TREE

- A hash tree or Merkle tree is a tree structure in which each leaf node is a hash of a block of data and each non-leaf node is a hash of its children. This results in a single hash called the Merkle root. If every node has two children, the tree is called a binary hash tree.



# MERKLE TREE

- Why use a Merkle tree?
  Why not hash all messages, append the hashed messages and then hash it all to get one root hash value.




- Bob get the root hash from a trusted source. If Alice wants to proof to Bob that $m_6$ is not tampered with, she needs to send message $m_6$ and all other hashed messages to Bob. Bob hashes message $m_6$, append all hashes to a single string and hash this string to get one root hash. Bob compares this new root hash with the trusted source root hash to check if message $m_6$ is not tampered with.

# MERKLE TREE

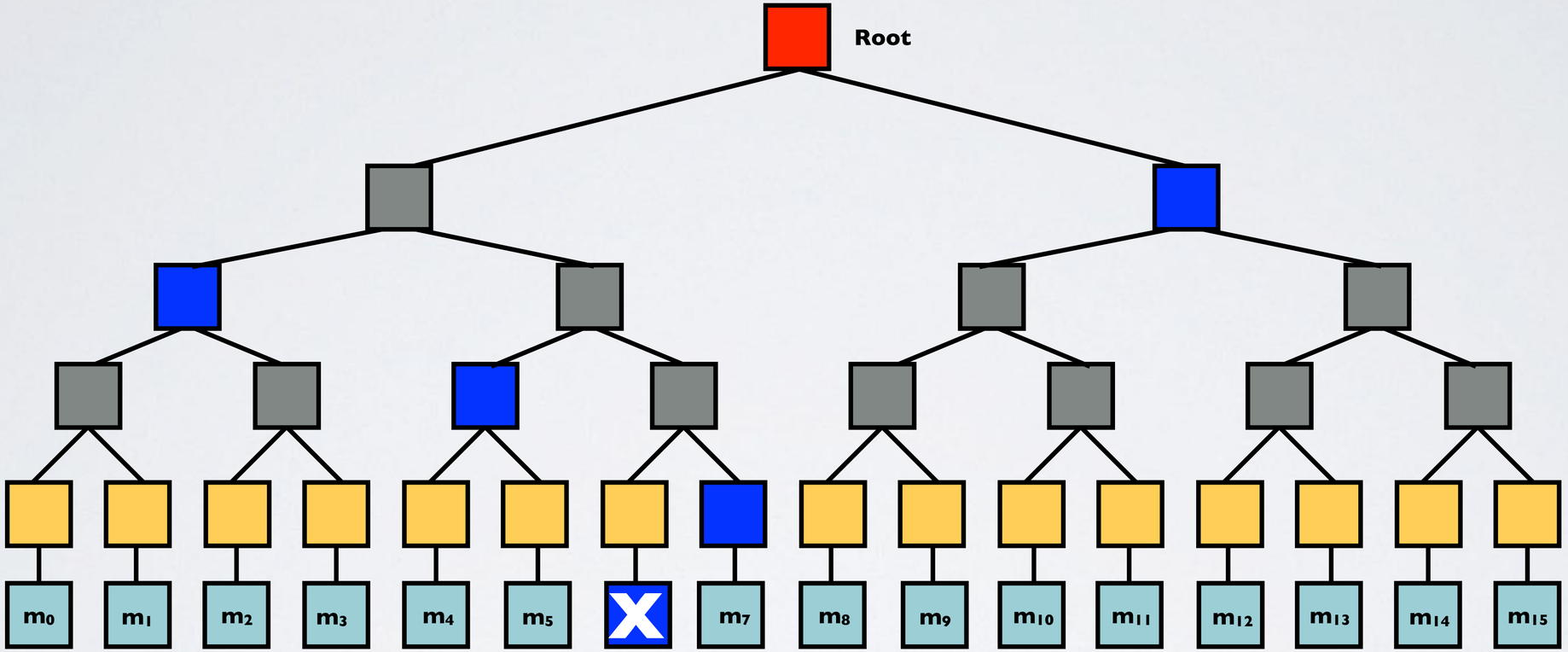
- In this example Alice has to provide 15 hashed values and the message $m_6$ to Bob to prove that message $m_6$ is not tampered with.

- A much better solution is using a Merkle tree.

- Again as before Bob gets the root hash from a trusted source. If Alice wants to proof that $m_6$ is not tampered with, she needs to send $m_6$ and 4 hashed values to Bob. With the received information Bob calculates the root hash value. Bob compares this root hash with the trusted source root hash to check if message $m_6$ is not tampered with.

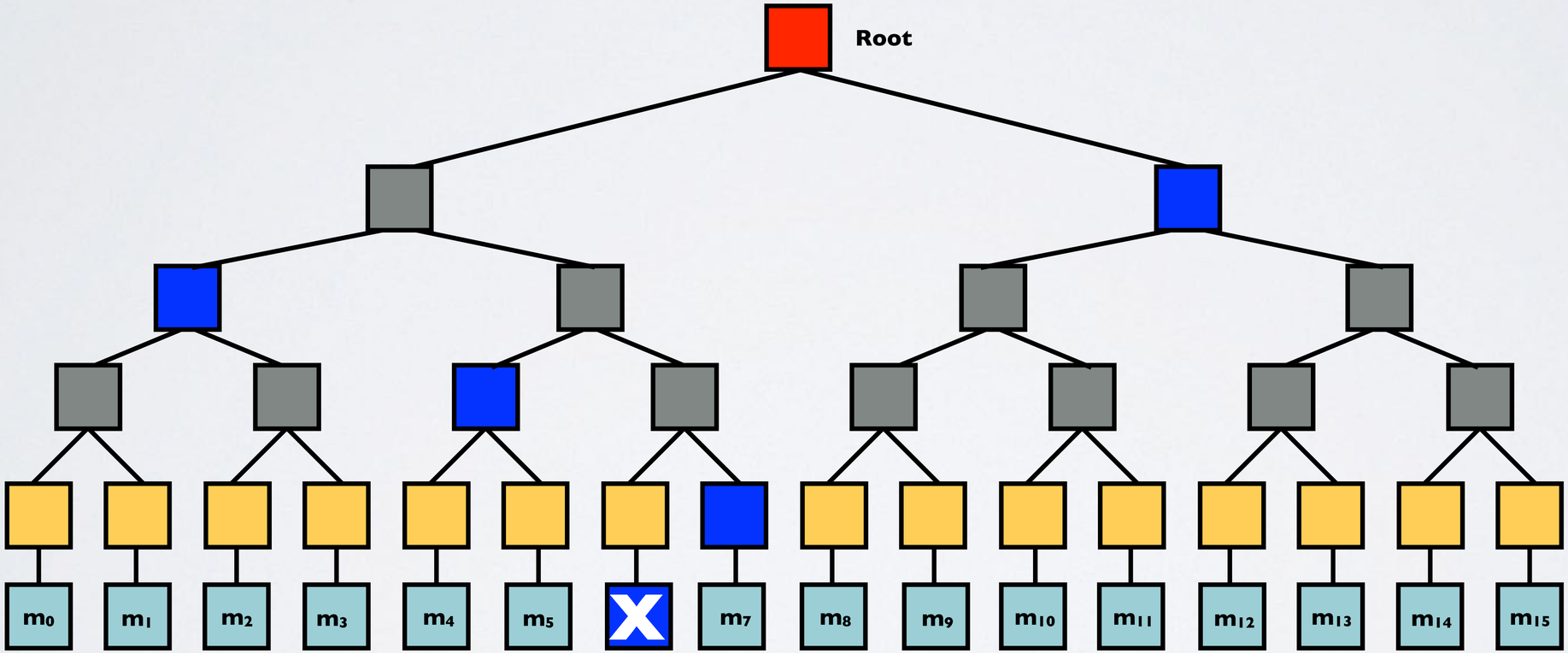
- In this example Alice only needs to provide 4 hashed values and the message $m_6$ to Bob to prove that message $m_6$ is not tampered with.

# MERKLE TREE



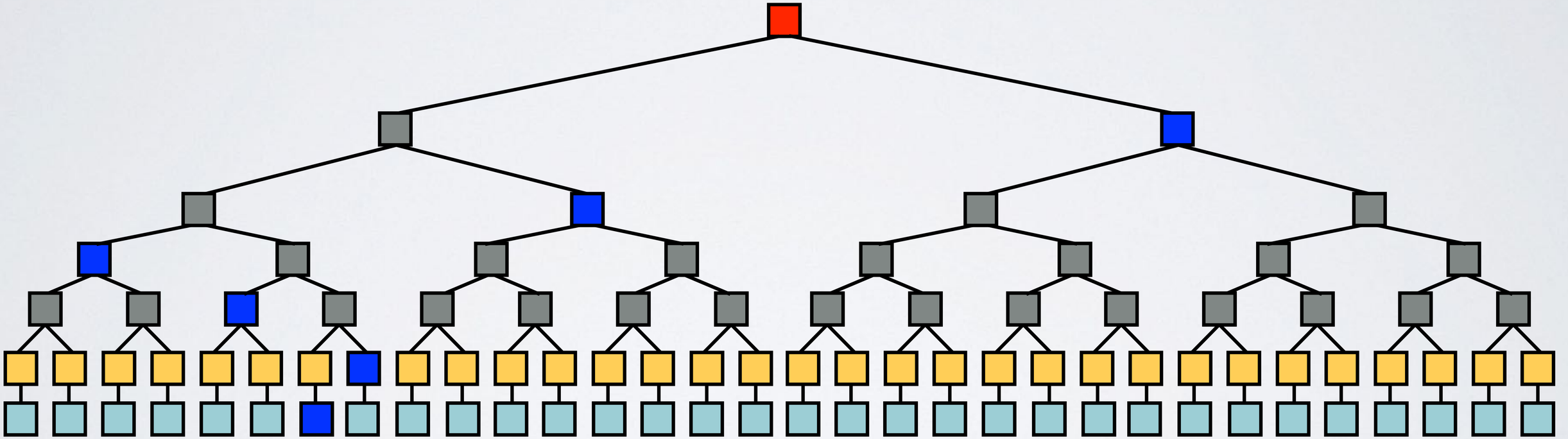
# MERKLE TREE

- Using a Merkle tree provides integrity and validity of your data using a small amount of data that a trusted authority has to maintain. This also means little memory / disk space is needed.



# MERKLE TREE

- If a Merkle tree has more leaves less hashed values are needed, in comparison to the number of leaves, to validate if a message is not tampered with.



# MERKLE TREE

- A perfect Merkle binary tree has the following properties:
  - The number of leaves is always $2^n$ (n=0, 1, 2, 3,..).
  - Each node has 0 or 2 children.
  - All leaves are on the same level.

- In a perfect binary tree the following formulas can be applied:

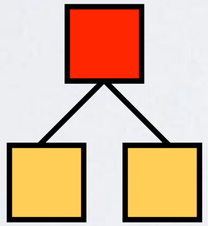
$$\textbf{Total number of leaves} = L = (N + 1) / 2$$
$$\textbf{Total number of nodes} = N = 2L - 1$$
$$\textbf{Total number of levels} = LV = \log_2(L) + 1$$
$$LV = (\ln(L) / \ln(2)) + 1$$

# MERKLE TREE: PERFECT BINARY TREE

Level 1

```
Number of leaves = L  = 1
Number of nodes  = N  = 1
Number of levels = LV = 1
```

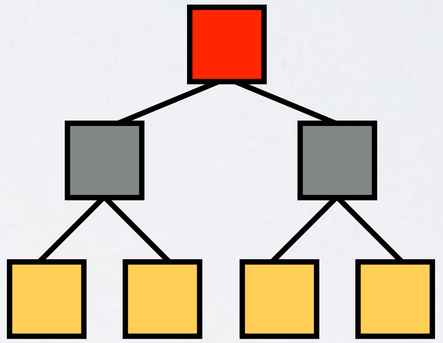
This Merkle tree has only one leaf.
This leaf is also the root.



Level 1

Level 2

```
Number of leaves = L  = 2
Number of nodes  = N  = 3
Number of levels = LV = 2
```



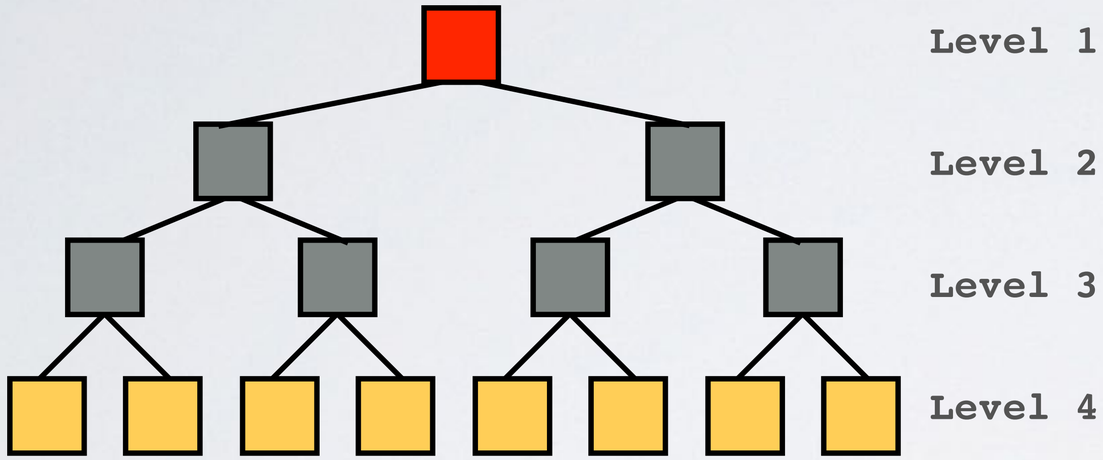
Level 1

Level 2

Level 3

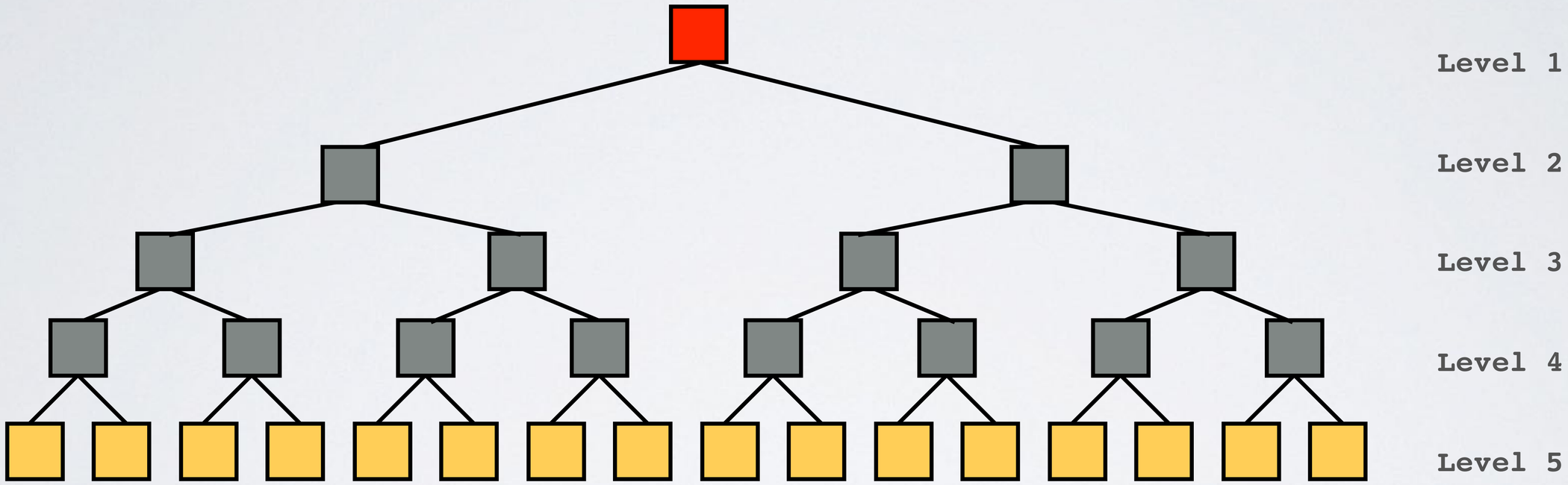
```
Number of leaves = L  = 4
Number of nodes  = N  = 7
Number of levels = LV = 3
```

# MERKLE TREE: PERFECT BINARY TREE




```
Number of leaves = L  = 8
Number of nodes  = N  = 15
Number of levels = LV = 4
```

# MERKLE TREE: PERFECT BINARY TREE




```
Number of leaves = L  = 16
Number of nodes  = N  = 31
Number of levels = LV = 5
```