Conclusion: Formally Defining a Blockchain

A decentralized immutable append-only public ledger

Cryptographic Hash Functions

- Takes any arbitrarily sized string as input
  Input M: The message
- **Fixed size output** (We typically use 256 bits in Blockchain)
  Output H(M): We call this as the message digest
- **Efficiently computable**

Cryptographic Hash Functions: Properties

- **Deterministic**
  Always yields identical hash value for identical input data
- **Collision-Free**
  If two messages are different, then their digests also differ
- **Hiding**
  Hide the original message; remember about the **avalanche effect**
- **Puzzle-friendly**
  Given *X* and *Y*, find out *k* such that $Y = H(X||k)$ - used to solve the mining puzzle in Bitcoin Proof of Work

Collision Free

- Hash functions are one-way; Given an x, it is easy to find $H(x)$. However, given an $H(x)$, **one cannot find** x
- It is **difficult to find** x and y, where $x \neq y$, but $H(x) = H(y)$
- Note the phrase **difficult to find**, collision is **not impossible**
- Try with randomly chosen inputs to find out a collision – but it takes too long

Hash Function – SHA256

- SHA256 is used in Bitcoin mining – to construct the Bitcoin blockchain
- Secure Hash Algorithm (SHA) that generates 256 bit message digest
- A part of SHA-2, a set of cryptographic hash functions designed by United States National Security Agency (NSA)

Step one: Appending bits

The first step involves preprocessing the input message to make it compatible with the hash function. It can be divided into two main substeps:

Padding bits

The total length of our message must be a multiple of 512. In this step, we append bits to the end of our message such that the final length of the message must be 64 bits less than a multiple of 512. The formula below depicts this step:

$$\boxed{m + p = (n * 512) - 64}$$

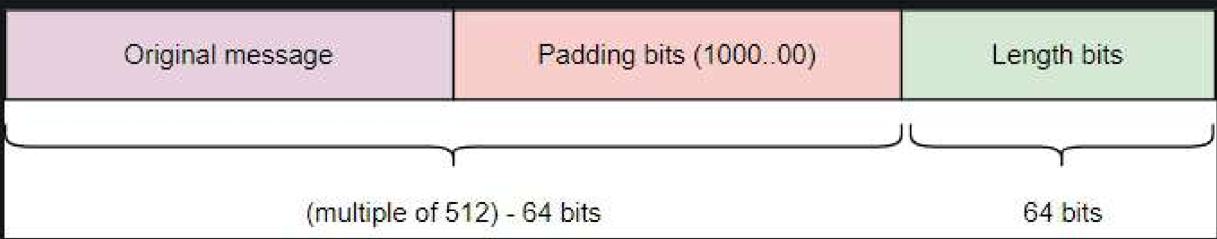
where m = length of the message, p = length of the padding, and n = a constant.

The first bit that we append is 1 followed by all 0 bits.

Length bits

Next, we take the modulus of the original message with 2^{32} to get 64 bits of data. Appending this to the padded message makes our processed message an exact multiple of 512.

The image below illustrates the final message after step one is completed.



The original message after preprocessing

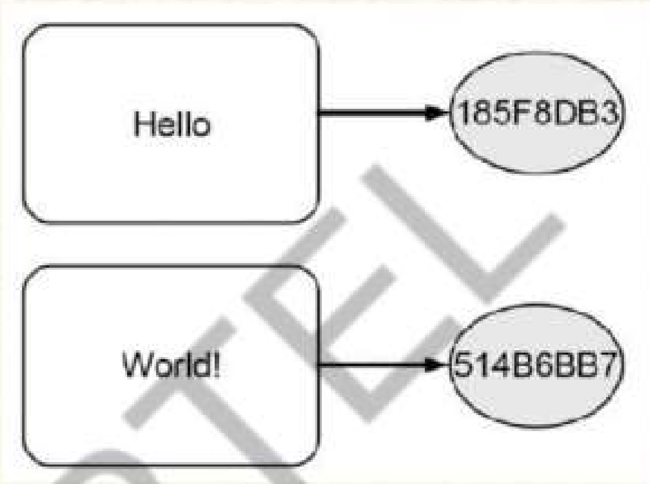
- Partition the message into $\boldsymbol{N}$ 512-bit blocks $\boldsymbol{M}^{(1)}, \boldsymbol{M}^{(2)}, \ldots, \boldsymbol{M}^{(N)}$
- Every 512 bit block is further divided into 32 bit sub-blocks $\boldsymbol{M}_{0}^{(i)}$, $\boldsymbol{M}_{1}^{(i)}, \ldots, \boldsymbol{M}_{15}^{(i)}$

SHA-256 Algorithm

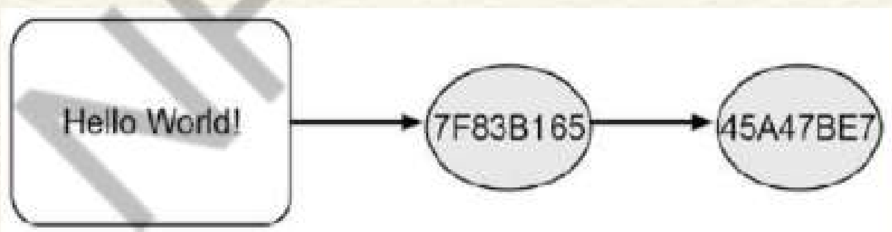
- The message blocks are processed one at a time
- Start with a fix initial hash value $H^{(0)}$
- Sequentially compute $H^{(i)} = H^{(i-1)} + C_{M^{(i)}}(H^{(i-1)})$; C is the SHA-256 *compression function* and + means mod 2^{32} addition. $H^{(N)}$ is the hash of M.

Types of Hashing

- Independent hashing



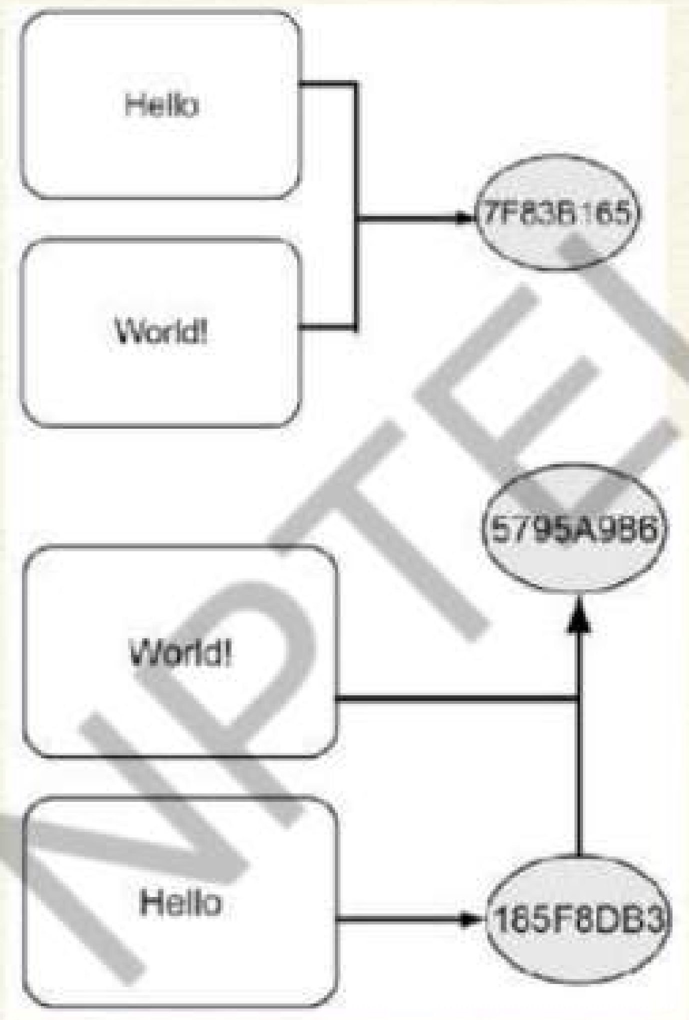
- Repeated hashing



Types of Hashing

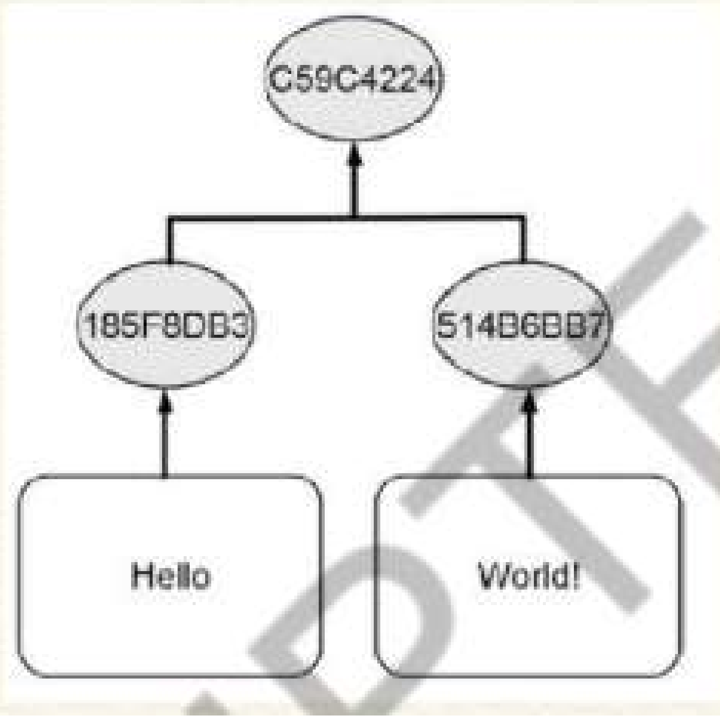
Combined hashing

Sequential hashing



Types of Hashing

Hierarchical hashing



Detect Tampering from Hash Pointers - Hashchain