

# Simple Blockchain Application in Swift

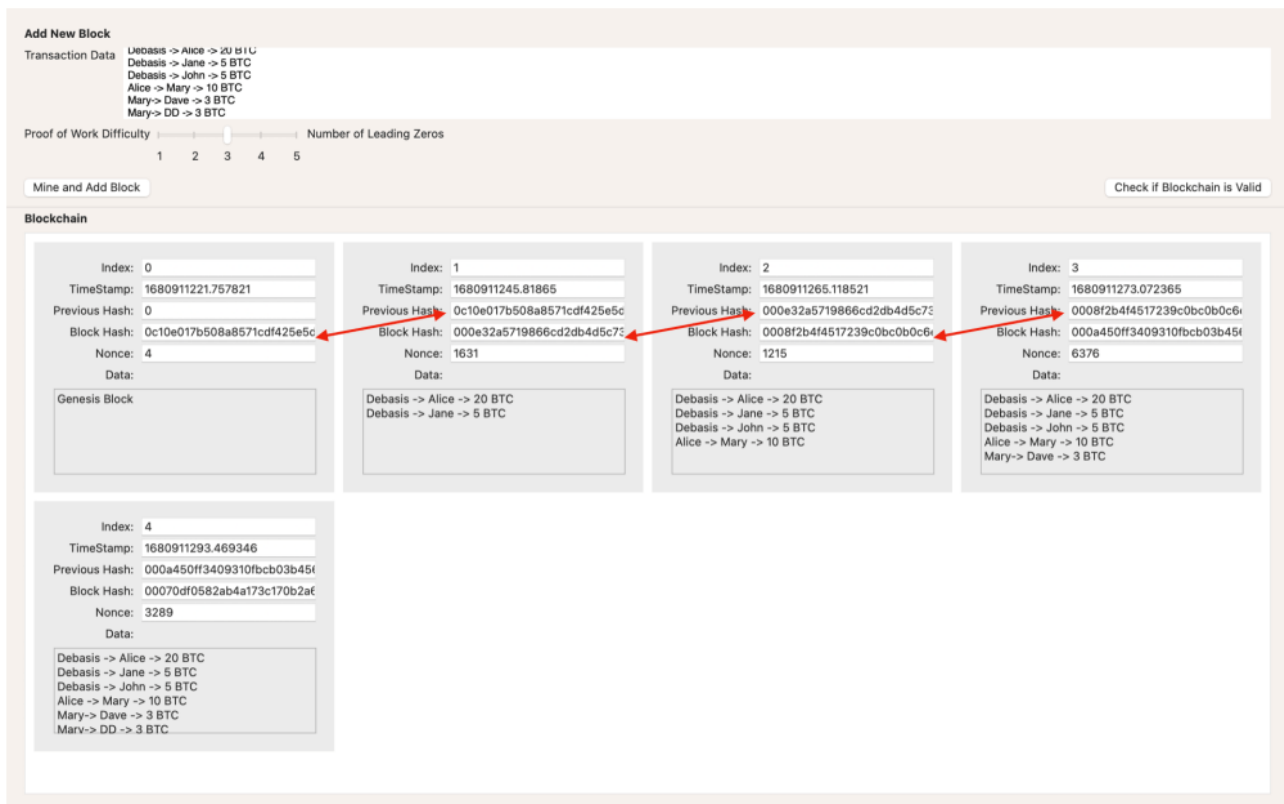
 [ddas.tech/simple-blockchain-application-in-swift/](https://ddas.tech/simple-blockchain-application-in-swift/)

April 8, 2023

Created By: Debasis Das (Apr 2023)

In this post, will create a simple blockchain application in Swift using CommonCrypto library with a variable degree of difficulty in the proof of work.

This application is for understanding the blockchain concept and is a very simple implementation.



The screenshot displays a web-based interface for a simple blockchain application. At the top, there's a section titled "Add New Block" with a "Transaction Data" field containing a list of transactions: Debasis -> Alice -> 20 BTC, Debasis -> Jane -> 5 BTC, Debasis -> John -> 5 BTC, Alice -> Mary -> 10 BTC, Mary -> Dave -> 3 BTC, and Mary -> DD -> 3 BTC. Below this is a "Proof of Work Difficulty" slider set to 3, with a "Number of Leading Zeros" field. There are two buttons: "Mine and Add Block" and "Check if Blockchain is Valid".

The main section, titled "Blockchain", shows a list of blocks. Each block contains the following information:

- Index
- TimeStamp
- Previous Hash
- Block Hash
- Nonce
- Data

The blocks shown are:

- Index: 0 (Genesis Block)
- Index: 1
- Index: 2
- Index: 3
- Index: 4

Red arrows indicate the chain of previous hashes: from Index 0 to Index 1, from Index 1 to Index 2, and from Index 2 to Index 3.

## Table of Contents

- [What is a Blockchain?](#)
- [Genesis Block](#)
- [Proof Of Work vs Proof of Stake](#)
- [Sample Application to Demonstrate the Blockchain App](#)
- [Block Sample Code](#)
- [Blockchain Sample Code](#)
- [User Interface to view the blockchain and trigger mining of new blocks](#)
- [Finally the App Delegate Class to get things rolling](#)

- [Download the Code](#)
- [References](#)

## What is a Blockchain?

---

A blockchain is a decentralized, distributed ledger technology that enables secure and transparent peer-to-peer transactions without the need for any intermediaries.

Each block in a blockchain contains a list of transactions and a unique cryptographic hash that links it to the previous block in the chain. The link between a block hash and the hash of the previous block creates an unbroken chain of blocks, with each block depending on the previous block for its validity. This mechanism ensures that once a block is added to the chain, it cannot be altered or deleted, making the blockchain tamper-proof and immutable.

The decentralized nature of blockchain means that there is no central authority controlling the network, and all participants have equal access to the same information.

Transactions on the blockchain are verified by network participants through a consensus mechanism, which ensures that all parties agree on the validity of each transaction.

## Genesis Block

---

Genesis block is the first block in a blockchain network, which is typically hardcoded into the protocol. It is the starting point of the blockchain and contains no previous block references since it is the first block ever created. The genesis block is usually created by the blockchain's creator or developers and is hardcoded with specific characteristics such as a unique hash, a timestamp, and other necessary data.

In addition to marking the beginning of the blockchain, the genesis block also establishes the initial parameters of the network, including the difficulty level of mining, the maximum supply of tokens, and other rules governing the network. The sample code provided here is for trying by modifying the difficulty level of the Proof of Work and Mine a new block

Add New Block

Transaction Data

Proof of Work Difficulty

1

2

3

4

5

Number of Leading Zeros

Mine and Add Block

Check if Blockchain is Valid

Blockchain

Index: 0

TimeStamp: 1680911221.757821

Previous Hash: 0

Block Hash: 0c10e017b508a8571cd1425e5c

Nonce: 4

Data:

Genesis Block

## Proof Of Work vs Proof of Stake

Proof of work (PoW) and proof of stake (PoS) are two different consensus algorithms used in blockchain technology to validate transactions and add new blocks to the blockchain. Here's a brief summary of the differences between the two:

1. Proof of work: In PoW, miners compete to solve complex mathematical puzzles using computational power to validate transactions and add new blocks to the blockchain. The first miner to solve the puzzle is rewarded with a block reward in cryptocurrency. PoW is energy-intensive and requires high computing power.
2. Proof of stake: In PoS, validators are selected based on the amount of cryptocurrency they hold in their wallet. Validators are chosen to validate transactions and add new blocks to the blockchain based on a random selection algorithm. Validators are incentivized to act honestly, as they stand to lose their stake if they act maliciously. PoS is less energy-intensive than PoW.

## Sample Application to Demonstrate the Blockchain App

In the below code we have defined a Block Structure with the following member properties

- Index of the block (Helps define the order of block addition to the block chain)
- timestamp – when the block was added to the blockchain
- data – List of transactions
- previousHash – reference to the hash of the previous block

- hash- hash of the current block
- nonce – a variable that we can modify and append to the end of the data to create a unique hash that satisfies the POW- Proof of Work requirement of the block.

In addition to the above properties we will define a couple of functions

- Block Initializer function
- Hash Generation function that runs in a while loop till the time the Proof of Work Requirement is not satisfied.
- Recalculate Hash – generates a hash for a given nonce value. This function will be used to validate the blockchain.

## **Block Sample Code**

---

```

// Blockchain.swift
// BlockchainApp
// Created by Debasis Das on 4/5/23.

import Foundation
import CommonCrypto

struct Block: Codable {
    let index: Int
    let timestamp: TimeInterval
    let data: String
    let previousHash: String
    var nonce: String
    let hash: String

    init(index: Int, timestamp: TimeInterval, data: String, previousHash:
String,powDifficulty:Int=1) {
        self.index = index
        self.timestamp = timestamp
        self.data = data
        self.previousHash = previousHash
        let tuple = Block.calculateHash(index: index, timestamp: timestamp, data:
data, previousHash: previousHash,powDifficulty: powDifficulty)
        self.hash = tuple.0
        self.nonce = tuple.1
    }

    static func recalculateHash(index: Int, timestamp: TimeInterval, data: String,
previousHash: String,nonce:String)->String{
        let message = "\(index)\(timestamp)\(data)\(previousHash)\(nonce)"
        if let messageData = message.data(using: .utf8) {
            var digest = [UInt8](repeating: 0, count: Int(CC_SHA256_DIGEST_LENGTH))
            messageData.withUnsafeBytes {
                _ = CC_SHA256($0.baseAddress, UInt32(messageData.count), &digest)
            }
            let hash = digest.map { String(format: "%02x", $0) }.joined()
            return hash
        }
        return ""
    }

    static func calculateHash(index: Int, timestamp: TimeInterval, data: String,
previousHash: String,powDifficulty:Int = 1) -> (String,String) {
        var nonce = 0
        var flag:Bool = true
        while(flag){
            nonce += 1
            let message = "\(index)\(timestamp)\(data)\(previousHash)\(nonce)"
            if let messageData = message.data(using: .utf8) {
                var digest = [UInt8](repeating: 0, count:
Int(CC_SHA256_DIGEST_LENGTH))
                messageData.withUnsafeBytes {

```

```

        _ = CC_SHA256($0.baseAddress, UInt32(messageData.count), &digest)
    }
    let hash = digest.map { String(format: "%02x", $0) }.joined()

    let subString = hash.prefix(powDifficulty)
    var valid = true
    for char in subString{
        if char != "0"{
            valid = false
        }
    }
    if valid{
        print(data, hash, nonce)
        flag = false
        return (hash, "\ (nonce)")
    }
}
}
return ("","0")
}
}

```

## Blockchain Sample Code

---

The blockchain code is rather simple with a member variable to store the array of blocks in the blockchain and the below functions

1. Block chain Initializer
2. Create Genesis Block/ First Block and adding to the network
3. Add Block to Block Chain function
4. Function to Validate Blockchain

```

class Blockchain {
    var chain: [Block] = []
    var pendingTransactions: [String] = []

    init() {
        createGenesisBlock()
    }

    func createGenesisBlock() {
        let genesisBlock = Block(index: 0, timestamp: Date().timeIntervalSince1970,
data: "Genesis Block", previousHash: "0",powDifficulty: 1)
        chain.append(genesisBlock)
    }

    func addBlock(data: String, powDifficulty:Int) {
        let previousBlock = chain.last!
        let block = Block(index: chain.count, timestamp:
Date().timeIntervalSince1970, data: data, previousHash:
previousBlock.hash,powDifficulty: powDifficulty)
        chain.append(block)
    }

    func validateChain() -> Bool {
        for i in 1..<chain.count {
            let block = chain[i]
            let previousBlock = chain[i-1]
            if block.hash != Block.recalculateHash(index: block.index, timestamp:
block.timestamp, data: block.data, previousHash: previousBlock.hash, nonce:
block.nonce){
                return false
            }
        }
        return true
    }
}

```

## User Interface to view the blockchain and trigger mining of new blocks

---

In the below code have used **NSCollectionView** to render the block chain. Every block in the block chain is represented as a **NSCollectionViewItem**

```

// BlockchainViewController.swift
// BlockchainApp
// Created by Debasis Das on 4/7/23.

import Cocoa
import Foundation

class BlockchainViewController: NSViewController, NSCollectionViewDataSource,
NSCollectionViewDelegate, NSCollectionViewDelegateFlowLayout {

    @IBOutlet weak var newBlockTransactionTextView: NSTextView!
    @IBOutlet weak var collectionView: NSCollectionView!
    @IBOutlet weak var powDifficulty: NSSlider!
    var blockchain: Blockchain!

    @IBAction func mineAndAddBlock(_ sender: Any){
        let tData = self.newBlockTransactionTextView.string
        if (tData != ""){
            self.blockchain.addBlock(data: tData, powDifficulty:
self.powDifficulty.integerValue)
            self.collectionView.reloadData()
        }
    }

    @IBAction func checkBlockchainValidity(_ sender: Any){
        let alert = NSAlert()
        if (self.blockchain.validateChain()){
            alert.messageText = "Valid"
            alert.informativeText = "Yes the block chain is Valid."
            alert.addButton(withTitle: "OK")
        } else{
            alert.messageText = "In-Valid"
            alert.informativeText = "Blockchain is not valid"
            alert.addButton(withTitle: "OK")
        }

        let response = alert.runModal()
        if response == .alertFirstButtonReturn {
            print("Do Nothing")
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        collectionView.dataSource = self
        collectionView.delegate = self
        collectionView.register(NSNib(nibNamed: "BlockItemView", bundle: nil),
forItemWithIdentifier: NSUserInterfaceItemIdentifier("BlockItemView"))
    }

    // MARK: - NSCollectionViewDataSource

```



```

    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection
section: Int) -> Int {
        //return items.count
        self.blockChain.chain.count
    }

    func collectionView(_ collectionView: UICollectionView,
itemForRepresentedObjectAt indexPath: IndexPath) -> UICollectionViewCellItem {
        let item = collectionView.makeItem(withIdentifier:
NSUserInterfaceItemIdentifier("BlockItemView"), for: indexPath)
        guard let itemView = item as? BlockItemView else {
            return item
        }

        if let block = self.blockChain.chain[indexPath.item] as? Block{
            itemView.indexTextField.stringValue = "\(block.index)"
            itemView.timestampTextField.stringValue = "\(block.timestamp)"
            itemView.previousHashTextField.stringValue = block.previousHash
            itemView.blockHashTextField.stringValue = block.hash
            itemView.nonceTextField.stringValue = block.nonce
            itemView.dataTextField.stringValue = block.data
        }
        return item
    }

    // MARK: - UICollectionViewDelegate
    func collectionView(_ collectionView: UICollectionView, layout
collectionViewLayout: UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) ->
NSSize {
        return NSSize(width: 337, height: 280)
    }

    func collectionView(_ collectionView: UICollectionView, layout
collectionViewLayout: UICollectionViewLayout, insetForSectionAt section: Int) ->
NSEdgeInsets {
        return UIEdgeInsets(top: 10, left: 10, bottom: 10, right: 10)
    }
}

```

## Finally the App Delegate Class to get things rolling

---

```
// AppDelegate.swift
// BlockchainApp
// Created by Debasis Das on 4/5/23.

import Cocoa

@main
class AppDelegate: NSObject, NSApplicationDelegate {

    @IBOutlet var window: NSWindow!
    var viewController:BlockChainViewController!

    func createConstraints(baseView: NSView, subView:NSView){
        subView.translatesAutoresizingMaskIntoConstraints = false
        let variableBindings = ["subView": subView]
        let vConstraints = NSLayoutConstraint.constraints(withVisualFormat: "V:|-0-[subView]-0-|",options: .alignAllLastBaseline, metrics: nil, views: variableBindings)
        let hConstraints = NSLayoutConstraint.constraints(withVisualFormat: "H:|-0-[subView]-0-|",options: .alignAllLastBaseline, metrics: nil, views: variableBindings)
        baseView.addConstraints(vConstraints)
        baseView.addConstraints(hConstraints)
    }

    func applicationDidFinishLaunching(_ aNotification: Notification) {
        let blockchain = Blockchain()
        self.viewController = BlockChainViewController(nibName:
"BlockChainViewController", bundle: .main)
        self.viewController.blockChain = blockchain

        self.window.contentView?.addSubview(self.viewController.view)
        self.createConstraints(baseView: self.window.contentView!, subView:
self.viewController.view)
    }
}
```

## Download the Code

---

<https://github.com/debasisdas1/BlockChainSwift.git>

## References

---

- <https://ethereum.org/en/>
- <https://en.wikipedia.org/wiki/Blockchain>