ECE 209 CA5
Zan Xie 205364923
Mar 20, 2024

The Apps Script code is capable to

1. Get the last column index as the reference.
2. Retrieve the hash value from the last block.
3. Generate a transaction block with the previous hash, timestamp, nonce=0, and transaction detail.
4. Include all contents in that block and pass to hashing.
5. Iterate nonce from 0 to nonstop until the hash starts with 4 zeros.
6. Update the nonce in that transaction block.
7. Expect the calculated hash and the hash from the SHA256() function in Google Sheets to be the same.

Result

1. My Own Blockchain: Compute a valid hash that starts with 4 zeros. The nonce = 2e2cc. The hash output from Apps Script matches the hash given by the SHA256() function
2. Class Blockchain: Retrieve hash value from my classmate's block#34 and successfully compute a valid hash that starts with 4 zeros. The nonce = 10622.

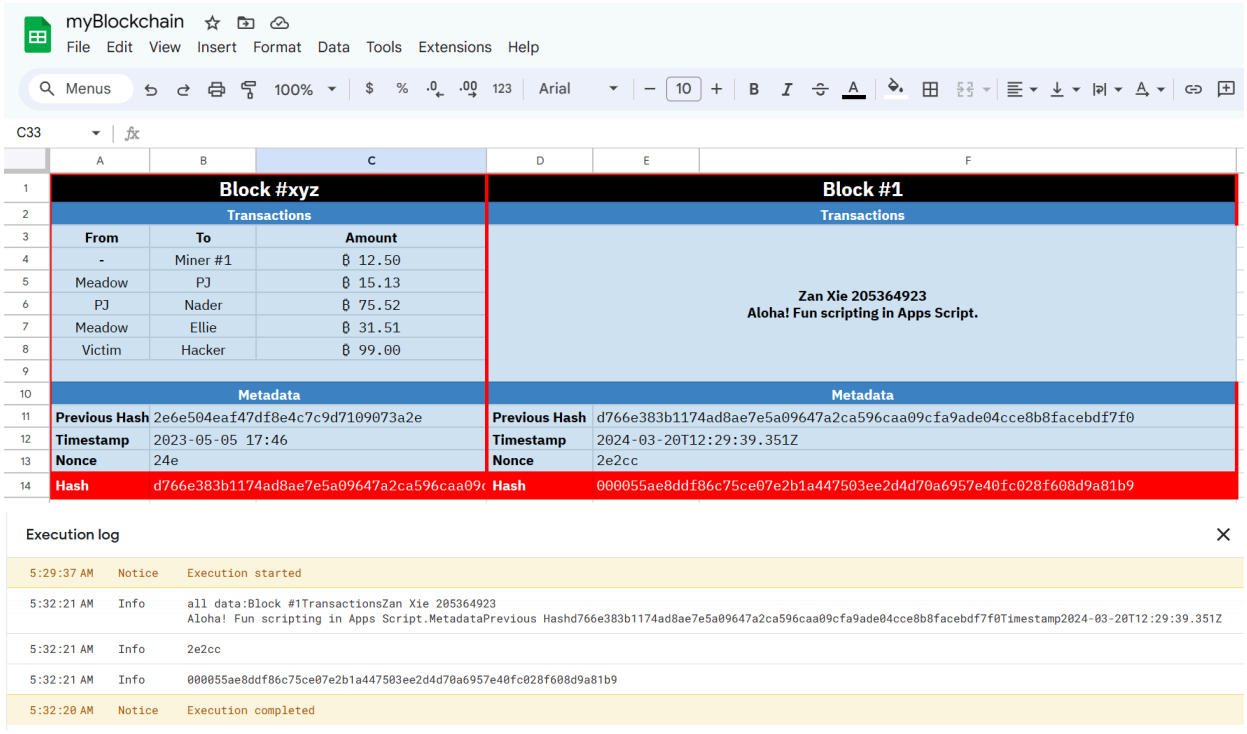
Python Project (Bonus Completed!!!)

1. The Python code imports a library "gspread" for direct access to Google Sheets.
2. Get the last non-empty column index in the worksheet.
3. Retrieve the hash value from the last block with last_col_idx as the reference index.
4. The SHA256() function used to compute the hash in the initial block is not recognizable by Python, therefore I recalculated the hash rather than retrieving it (retrieving leads to error: #NAME?).
5. Create an empty 2d array and load in the previous hash, transaction details, timestamp, etc.
6. Pass data to the proof_of_work() function which iterates nonce from 0 to nonstop until the hash starts with 4 zeros.
7. Update the destination worksheet range with found nonce and hash.

Result

1. The worksheet begins with only the initial block. The initial block is assigned to be Block #0 instead of Block #xyz. Additionally, I copy the cell format over, maintaining format consistency.
2. I ran Python code twice and thus generated two blocks. Each block connects to its previous block. This work shows that the Python code can generate new blocks on its own.

My Own Blockchain



myBlockchain

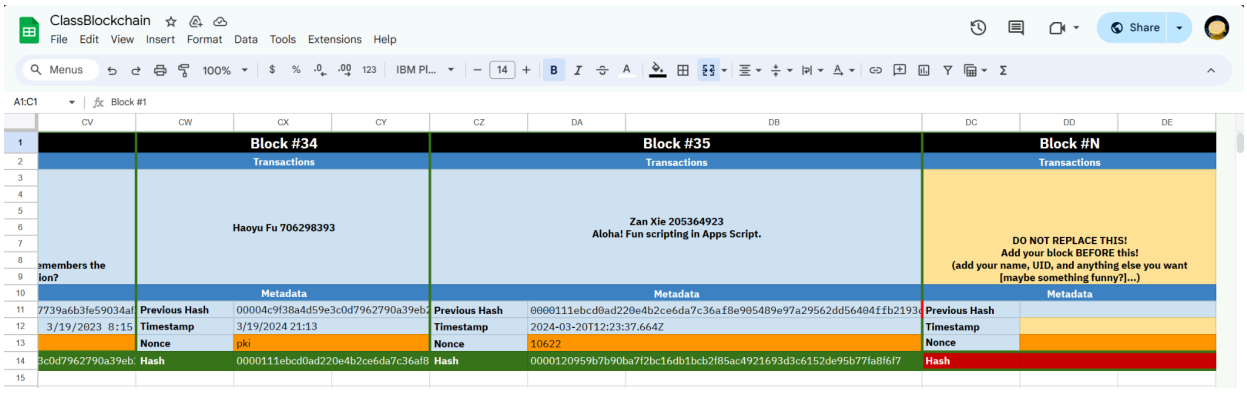
File Edit View Insert Format Data Tools Extensions Help

Block #xyz			Block #1	
Transactions			Transactions	
From	To	Amount		
-	Miner #1	₿ 12.50		
Meadow	PJ	₿ 15.13		
PJ	Nader	₿ 75.52		
Meadow	Ellie	₿ 31.51		
Victim	Hacker	₿ 99.00		
			Zan Xie 205364923 Aloha! Fun scripting in Apps Script.	
Metadata			Metadata	
Previous Hash	2e6e504eaf47df8e4c7c9d7109073a2e		Previous Hash	d766e383b1174ad8ae7e5a09647a2ca596caa09cfa9ade04cce8b8facebdf7f0
Timestamp	2023-05-05 17:46		Timestamp	2024-03-20T12:29:39.351Z
Nonce	24e		Nonce	2e2cc
Hash	d766e383b1174ad8ae7e5a09647a2ca596caa09(		Hash	000055ae8ddf86c75ce07e2b1a447503ee2d4d70a6957e40fc028f608d9a81b9

Execution log

5:29:37 AM	Notice	Execution started
5:32:21 AM	Info	all data:Block #1TransactionsZan Xie 205364923 Aloha! Fun scripting in Apps Script.MetadataPrevious Hashd766e383b1174ad8ae7e5a09647a2ca596caa09cfa9ade04cce8b8facebdf7f0Timestamp2024-03-20T12:29:39.351Z
5:32:21 AM	Info	2e2cc
5:32:21 AM	Info	000055ae8ddf86c75ce07e2b1a447503ee2d4d70a6957e40fc028f608d9a81b9
5:32:20 AM	Notice	Execution completed

Class Blockchain



ClassBlockchain

File Edit View Insert Format Data Tools Extensions Help

Block #34			Block #35		Block #N	
Transactions			Transactions		Transactions	
			Haoyu Fu 706298393		Zan Xie 205364923 Aloha! Fun scripting in Apps Script.	
					DO NOT REPLACE THIS! Add your block BEFORE this! (add your name, UID, and anything else you want [maybe something funny?]...)	
Metadata			Metadata		Metadata	
7739a6b3fe59034af	Previous Hash	00004c9f38a4d59e3c0d7962790a39eb2	Previous Hash	0000111ebcd0ad220e4b2ce6da7c36af8e905489e97a29562dd56404ffb2193	Previous Hash	
3/19/2023 8:15	Timestamp	3/19/2024 21:13	Timestamp	2024-03-20T12:23:37.664Z	Timestamp	
	Nonce	pki	Nonce	10622	Nonce	
3c0d7962790a39eb	Hash	0000111ebcd0ad220e4b2ce6da7c36af8	Hash	0000120959b7b90ba7f2bc16db1bcb2f85ac4921693d3c6152de95b77fa8f6f7	Hash	

Python Project Blockchain

myBlockchainPython — Saved to Drive

File Edit View Insert Format Data Tools Extensions Help

Block #0		
Transactions		
From	To	Amount
-	Miner #1	฿ 12.50
Meadow	PJ	฿ 15.13
PJ	Nader	฿ 75.52
Meadow	Ellie	฿ 31.51
Victim	Hacker	฿ 99.00
Metadata		
Previous Hash	2e6e504eaf47df8e4c7c9d7109073a2e	
Timestamp	2023-05-05 17:46	
Nonce	24e	
Hash	4e1c28a33401fe78396cc252d290f18ef89fa11	

Fig1. The initial setup

myBlockchainPython

File Edit View Insert Format Data Tools Extensions Help

Block #0			Block #1			Block #2		
Transactions			Transactions			Transactions		
From	To	Amount	From	To	Amount	From	To	Amount
-	Miner #1	฿ 12.50	-	Miner #1	฿ 12.50	-	Miner #1	฿ 12.50
Meadow	PJ	฿ 15.13	Stan	Bob	฿ 9.90	Stan	Bob	฿ 9.90
PJ	Nader	฿ 75.52						
Meadow	Ellie	฿ 31.51						
Victim	Hacker	฿ 99.00						
Metadata			Metadata			Metadata		
Previous Hash	2e6e504eaf47df8e4c7c9d7109073a2e		Previous Hash	4e1c28a33401fe78396cc252d290f18ef89		Previous Hash	0000ea4b0bcf46f11fc3cd8cfc96a570b9b4a	
Timestamp	2023-05-05 17:46		Timestamp	03/20/2024 18:23:07		Timestamp	03/20/2024 18:24:06	
Nonce	24e		Nonce	b450		Nonce	1dda	
Hash	4e1c28a33401fe78396cc252d290f18e		Hash	0000ea4b0bcf46f11fc3cd8cfc96a570b9b4		Hash	0000fd253f2d8a6c54a280a04eb40c5902234	

Fig2. The blockchain

Apps Script Code

```
function SHA256 (input) {
  var digest = Utilities.computeDigest(Utilities.DigestAlgorithm.SHA_256, input);
  var output = "";
  for (i = 0; i < digest.length; i++) {
    var h = digest[i];
    if (h < 0) { h += 256; }
    if (h.toString(16).length == 1) { output += '0'; }
    output += h.toString(16);
  }
  return output;
}

// proof of work function
function proofOfWork(data) {
  var nonce = -1;
  var hash = ""
  var difficulty = "0000"; // You can adjust the difficulty here
  do {
    nonce++;
    nonce_str = nonce.toString(16);
    hash = SHA256(data + "Nonce" + nonce_str);
  } while (!hash.startsWith(difficulty));
  return {
    nonce: nonce_str,
    hash: hash
  }
}

// attach new block
function addBlock() {
  var sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();

  // the most recent block
  var lastColumn = sheet.getLastColumn(); // retrieve the last column the previous block
  var range = sheet.getRange(1, lastColumn - 4, 14, 3); // capture the most recent block
  var lastBlockData = range.getValues();
  var pre_hash_val = lastBlockData[13][1]; // retrieve previous hash value

  // construct block content
  var header_text = "Block #1";
  var content_text = "Zan Xie 205364923\nAloha! Fun scripting in Apps Script.";
  var pre_hash = pre_hash_val.toString();
  var timestamp = new Date().toISOString();

  // write into cells
  var startColumn = lastColumn - 1;
  var header_range = sheet.getRange(1, startColumn, 1, 1);
  var content_range = sheet.getRange(3, startColumn, 1, 1);
  var pre_hash_range = sheet.getRange(11, startColumn+1, 1, 1);
  var time_range = sheet.getRange(12, startColumn+1, 1, 1);
  header_range.setValue(header_text);
  content_range.setValue(content_text);
  pre_hash_range.setValue(pre_hash);
  time_range.setValue(timestamp);

  // search for a proper nonce
  data_range = sheet.getRange(1, startColumn, 12, 3);
  var data = data_range.getValues();
  var allData = "";
  for (var i = 0; i < data.length; i++) {
    for (var j = 0; j < data[i].length; j++) {
      allData += String(data[i][j]);
    }
  }
  var nonce = proofOfWork(allData).nonce;
  var cur_hash = proofOfWork(allData).hash;
  Logger.log("all data:" + allData);
  Logger.log(nonce);
  Logger.log(cur_hash);

  // write nonce into the cell
  var nonce_range = sheet.getRange(13, startColumn+1, 1, 1);
  nonce_range.setValue(nonce);

}
```

Python Code

ca5

March 20, 2024

1 ECE 209AS CA5

Zan Xie 205364923

[ ]:
```
# library
!pip install gspread
import gspread
from google.colab import auth
from google.auth import default
auth.authenticate_user()

import re
import hashlib
import datetime
```

[132]:
```
# compute SHA256 hash
def SHA256(input_string):
  return hashlib.sha256(input_string.encode()).hexdigest()

# proof of work function
def proof_of_work(data, difficulty='0000'):
  nonce = 0
  while True:
    nonce_str = hex(nonce)[2:]
    hash_result = SHA256(data + nonce_str)
    if hash_result.startswith(difficulty):
      return nonce_str, hash_result
    nonce += 1

# convert column index to letter notation
def index_to_letter(index):
  letters = ''
  while index > 0:
    index, remainder = divmod(index - 1, 26)
    letters = chr(65 + remainder) + letters
  return letters
```

[134]:
```
# load in worksheet
file_name = 'myBlockchainPython'
creds, _ = default()
gc = gspread.authorize(creds)
worksheet = gc.open(file_name).sheet1

all_values = worksheet.get_all_values()
last_col = len(all_values[0])

# the most recent block hash
# hardcode the most recent block's data and get its hash instead of retriving␣
 ↪it using get_value()
# detail explaination in project report
pre_block_index = int(worksheet.cell(1, last_col - 2).value.split('#')[1])
if (pre_block_index == 0):
  data = 'Block #0TransactionsFromToAmount-Miner #112.5MeadowPJ15.13PJNader75.
 ↪52MeadowEllie31.51VictimHacker99MetadataPrevious␣
 ↪Hash2e6e504eaf47df8e4c7c9d7109073a2e        Timestamp45051.
 ↪7404976852Nonce24e'
  pre_hash = SHA256(data)
else:
  pre_hash = worksheet.cell(14, last_col - 1).value

print('pre_block_idx:', pre_block_index)
print('pre_hash:',pre_hash)
```

```
pre_block_idx: 1
pre_hash: 0000ea4b0bcf46f11fc3cd8cfc96a570b9b4a6b710c061f67b4bbc673b661d3c
```

[135]:
```
# generate a transcation block
start_col = index_to_letter(last_col + 1)
end_col = index_to_letter(last_col + 3)
block_range = f"{start_col}1:{end_col}14"
block_data = [['' for _ in range(3)] for _ in range(14)]

# transcation record, minic real-transcation
pay = ['-', 'Stan']
receive = ['Miner #1', 'Bob']
amount = ['12.50', '9.90']

# datetime
block_index = pre_block_index + 1
now = datetime.datetime.now()
format_datetime = now.strftime("%m/%d/%Y %H:%M:%S")

# Set values in the block
block_data[0][0] = 'Block #' + str(block_index)
```

```
block_data[1][0] = 'Transactions'
block_data[2][0] = 'From'
block_data[2][1] = 'To'
block_data[2][2] = 'Amount'

# transcation content
trans_idx = 3
trans_limit = 7
for x,y,z in zip(pay, receive, amount):
  if (trans_idx <= trans_limit):
    block_data[trans_idx][0] = x
    block_data[trans_idx][1] = y
    block_data[trans_idx][2] = z
  trans_idx+=1

# metadata section
block_data[9][0] = 'Metadata'
block_data[10][0] = 'Previous Hash'
block_data[10][1] = pre_hash
block_data[11][0] = 'Timestamp'
block_data[11][1] = format_datetime
block_data[12][0] = 'Nonce'
block_data[13][0] = 'Hash'

# join block contents togeter and pass to hashing
allData = ''.join(''.join(row) for row in block_data)
print(allData)

nonce, hash_val = proof_of_work(allData)
print('nonce:',nonce)
print('hash:',hash_val)

# write nonce and hash into cell
block_data[12][1] = nonce
block_data[13][1] = hash_val

# Update the worksheet
worksheet.update(range_name=block_range, values=block_data)
```

```
Block #2TransactionsFromToAmount-Miner #112.50StanBob9.90MetadataPrevious Hash00
00ea4b0bcf46f11fc3cd8cfc96a570b9b4a6b710c061f67b4bbc673b661d3cTimestamp03/20/202
4 18:24:06NonceHash
nonce: 1dda
hash: 0000fd253f2d8a6c54a280a04eb40c5902234d66a697132e6c543af258e29907
```

[135]:
```
{'spreadsheetId': '15aRaxgSMDKUKe0s4ND1V1UarUc7wQZYgvm27BEL0p9s',
 'updatedRange': "'Block Structure'!G1:I14",
```

```
'updatedRows': 14,
'updatedColumns': 3,
'updatedCells': 42}
```