The Application of Graph and Number Theory for Data Security: Blockchain

Kenneth Ezekiel Suprantoni - 13521089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13521089@mahasiswa.itb.ac.id

***Abstract*—Graph and Number Theory's Application in the modern world is immense, and one of the groundbreaking inventions, especially for Data Security, made using the fundamentals of graph and number theory, is Blockchain. Blockchains offers us a way of transparency, yet secure way of keeping data, as its main theme is decentralization, furthermore, it provides a way of constant data that is easy to trace back. But how does the two theories combine exactly to make such a masterpiece as the blockchain?**

***Keywords*—Graph Theory, Number Theory, Data Security, Blockchain**

I. Introduction

Data, as defined by the Merriam-Webster dictionary, is a factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation, while data security, as defined by IBM, is the practice of protecting digital information from unauthorized access, corruption, or theft throughout its entire lifecycle. The main focus of this paper is the protection of data from corruption or manipulation, which introduces the concept of blockchain. Blockchain, most common in the field of economic-technology, is a way to store data that focuses primarily on the protection of the data from corruption or modification, its largest use-case is the cryptocurrency blockchain, where it is used to store the transactional data of the currency to keep track of the current flow of the currency, and the balance of users. In short, its implementation is done by storing a fixed amount of data in a block, hashing the data of the block, and storing the hash on the next block, which will include the previous block's hash as a data that will be hashed with its content.

Hashing is a concept of cryptography, which maps a data into a fixed size by using a hash function or a hash algorithm. One of the most popular hashing algorithms is the SHA-2 or Secure Hash Algorithm 2, which hashes a data into a cryptographic hash, or also often referenced to as a "signature", which is an almost perfectly unique string of characters that is generated from the data. This hashing concept can then be the guarantee that an arbitrary orientation of data, is not corrupted or modified, as it will have an almost perfectly unique "signature" for each of the unique data orientation. This concept is implemented in a blockchain, where each block will have a fixed amount of data, and when it is hashed, its content can't be modified anymore. The "chain" of blocks, where the next block will include the hash of the previous block will also be a reinforcement of the security, as there is a possibility to find the identic hash of two different data given an arbitrary line of input to the data, see *Reference 3: What is Proof of Work in Blockchain*. The "chain" will then guarantee the purity of the older block, as modifying the older block will result in an invalid hash for its next block, all the way into the last block, thus the longer the blockchain, the safer the data stored gets.

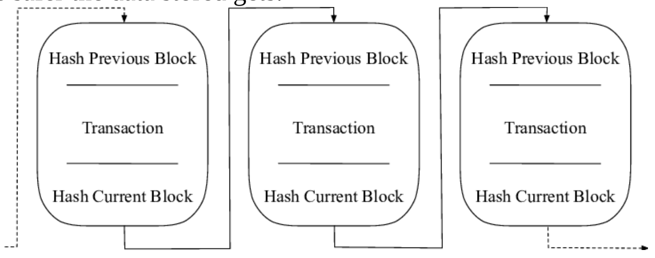


Fig. 1.1 Visual representation of a blockchain
Source: https://www.researchgate.net/figure/Visual-representation-of-three-blocks-of-a-Blockchain-Each-block-references-the-previous_fig4_328377682

This paper will focus on how blockchain is implemented based on graph theory and number theory, building from the basis to its application, to how its application can be implemented to reach the blockchain.

II. Theoretical Basis

1. Graph

1.1 Graph Definition and Types

Graphs are a way of representing how discrete objects connect with each other, or in other words, their relations. The objects, which is called vertices or nodes, can be connected to each other by an edge, representing its relation. Formally, a graph G is defined by a tuple of non-empty vertex set V, and edge set E:

$$G = (V, E)$$

Where for a set of *n* objects, the set V will contain:

$$V = \{v_1, v_2, \dots, v_n\}$$

And for a set of m relations, the set E will contain:

$$E = \{e_1, e_2, \dots, e_m\}$$

Graphs can also be of two types, simple graph, and unsimple-graph, and what differentiate the two is the latter contains a multi-edge, which is defined by two nodes having two or more edges to each other, and a loop, which is defined by a node having one or more edge to itself. The unsimple-graph can also be of two types, a multi-graph, a graph that contains a multi-edge, and a pseudo-graph, a graph that contains a loop.

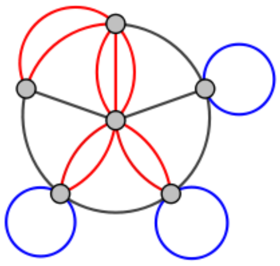


Fig 2.1 Example of an unsimple-graph
Source: https://en.wikipedia.org/wiki/Multigraph

By orientation, a graph can also be an undirected graph, having no directions on its edges, or a directed graph, the directions will represent a one-sided relation.

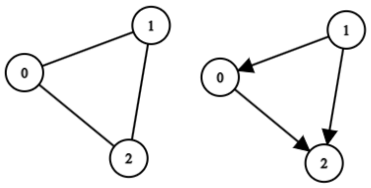


Fig 2.2 Example of directed and undirected graph
Source: Generated using https://csacademy.com/app/graph_editor/

1.2 Graph Terminology

- Adjacence
  A vertex is called adjacent to another vertex if it is directly connected to it.
- Incidence
  A vertex is called incident with an edge if the edge connects to the vertex.
- Isolated Vertex
  A vertex is called isolated if it has no edge connected to it.
- Null Graph
  A Null graph is a graph that contains no edges.
- Degree
  The degree of a vertex is the sum of how many edges are connected to the vertex or how many edges are incident with the vertex. The sum of all degrees in a graph will always be even, with a quantity of two times the sum of all edges. Which then can be derived that in a Graph G, the count of every vertex with odd degrees will be even.
- Path
  A finite or infinite sequence of edges which joins a sequence of vertices which are all distinct. A path with length n is a traversal sequence of vertex and edge $\{v_0, e_1, v_1, e_2, \ldots, v_{n-1}, e_n, v_n\}$.
- Circuit
  A circuit is a path that ends on the same vertex that it started.
- Connection
  A vertex v_1 and vertex v_2 is said to be connected if there is a path from v_1 to v_2. A Graph is said to be connected if for every pair of vertices in the graph, there is a path between the pair. If a graph is not connected, then it is said to be disconnected graph. For directed graphs, vertices u and v are said to be strongly connected if there is a directed path from u to v and also from v to u, if it is not strongly connected but the two vertices are connected in the undirected graph of the graph, then vertices u and v are said to be weakly connected.
- Subgraph
  A graph $G_1 = \{V_1, E_1\}$ is said to be a subgraph of graph $G = \{V, E\}$ if $V_1 \subseteq V$ and $E_1 \subseteq E$. A graph $G_2 = \{V_2, E_2\}$ is said to be a complement of subgraph G_1 if $V_2 \subseteq V$ and $E_2 = E - E_1$. The component of a graph is the maximum amount of connected subgraph in a graph G.
- Spanning Subgraph
  A spanning subgraph is a subgraph which has every vertex of the graph, graph $G_1 = \{V_1, E_1\}$ is said to be a spanning subgraph of graph $G = \{V, E\}$ if $V_1 = V$ and $E_1 \subseteq E$.
- Cut-Set
  A cut-set of a connected graph G is a set of edges that if discarded, results in graph G being disconnected, thus a cut-set always results in 2 components.
- Weighted Graph
  A weighted graph is a graph that has a weight in every edges.

1.3 Special Graphs

- Complete Graph (K_n)
  A complete graph is a simple graph that for each vertex, it has a connection to every vertex in the graph. In a complete graph consisting of n vertices, the sum of all edges in the graph will be $\Sigma|E| = \frac{n}{2}(n-1)$.

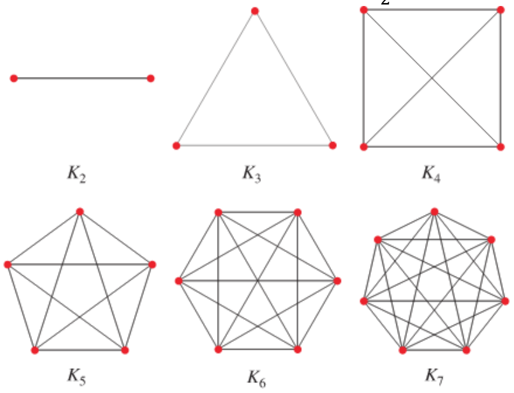


Fig. 1.3.1 Visualization of complete graphs
Source:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fmathworld.wolfram.com%2FCompleteGraph.html&psig=AOvVaw2E7RPV6ypTZQf9I39t7wUc&ust=1670742951789000&source=images&cd=vfe&ved=0CBIQjhxqFwoTCIDwtNPA7vsCFQAAAAAdAAAAABAE

- Circle Graph (C_n)
  A circle graph is a simple graph that has a degree of 2 for each vertex.

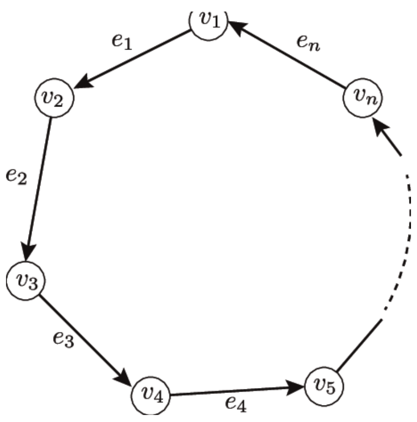


Fig. 1.3.2 Visualization of circle/cycle graphs
Source: https://www.researchgate.net/figure/A-cycle-graph-C-n-with-n-vertices-and-edges_fig1_363128491

- Regular Graph (R_r)

  A regular graph is a graph where for each vertex in the graph, it has the same number of degrees as all the other vertices in the graph. A regular graph with the degree of r has a degree of r for every vertex in the graph. The number of edges of a regular graph are $\Sigma|E| = \frac{n.r}{2}$, where n is the number of vertices in the graph.

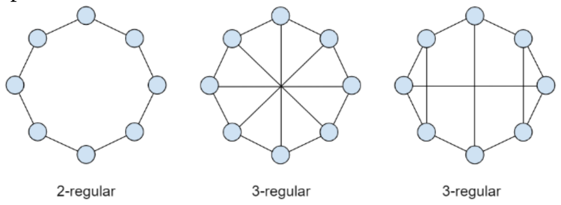


Fig. 1.3.3 Visualization of regular graphs
Source: https://thomasvilhena.com/2020/05/data-replication-in-random-regular-graphs

- Bipartite Graph ($G(V_1, V_2)$)

  A bipartite graph is a graph that for the set of vertices V, there can be two subsets of V, V_1 and V_2, which for every edge in the graph, the edge connects a vertex from V_1 to a vertex from V_2.

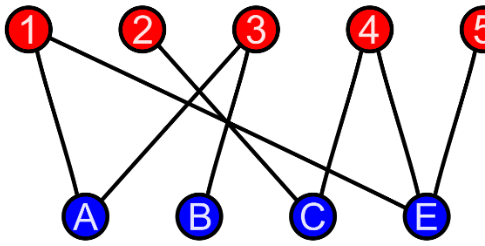


Fig. 1.3.4 Visualization of bipartite graphs
Source: https://en.wikipedia.org/wiki/Bipartite_graph

2. Number Theory

Number theory is the branch of mathematics that focuses on the study of integer and integer-valued function. By definition, integers are numbers that has no fractions, zero, the set of all positive natural number, or negative integer numbers. For an integer a and an integer b, a divides b if and only if there exist an integer c so that $b = ac$, notated as $a \mid b, c \in \boldsymbol{Z}, a \neq 0$.

2.1 Euclidean Theorem/Division

Let m, n be an integer where $n > 0$. If m is divided by n then the result of the division is q $(quotient)$ and the rest is r $(remainder), so\ that$

$$m = nq + r, \quad 0 \le r < n$$

2.2 Greatest Common Divisor

The greatest common divisor of two integers, a and b is the largest integer d where, $d \mid a$ and $d \mid b$, thus $GCD(a, b) = d$.

2.3 Euclidean Algorithm

Let m and n be a nonnegative integer with $m \geq n$. Let $r_0 = m, r_1 = n$. With continuous division:

$$r_0 = r_1 q_1 + r_2, \quad 0 \le r_2 < r_1$$
$$r_1 = r_2 q_2 + r_3, \quad 0 \le r_3 < r_2$$
$$\cdots$$
$$r_{n-2} = r_{n-1} q_{n-1} + r_n, \quad 0 \le r_n < r_{n-1}$$
$$r_{n-1} = r_n q_n + 0$$

as for every $m = nq + r$, $0 \le r < n$, $GCD(m, n) = GCD(n, r)$, thus:

$$GCD(m, n) = GCD(r_0, r_1) = GCD(r_1, r_2) = \cdots$$
$$= GCD(r_{n-1}, r_n) = GCD(r_n, 0) = r_n$$

2.4 Modulo Arithmetic

For every $m = nq + r$ equation, it can be described a modulo operations $m \bmod n = r, 0 \le r < m$, thus the mod operation will return the remainder of the number if divided by the $modulus$ (n).

2.5 Congruence

Given an integer $n > 1$, where n is said to be the modulus, two integers a and b are said to be congruent in the modulo of n, if $a \bmod n = r = b \bmod n$, which can be notated by $a \equiv b \pmod n$. Which can be rewritten as the equation $a = kn + b$, because b need not be the remainder of a divided by n, the equation can be broken down to:

$$a = pn + r \qquad (1)$$
$$b = qn + r \qquad (2)$$
$$0 \le r < n$$

Thus, by subtracting equation (1) and (2):

$$a - b = pn - qn$$
$$a - b = kn, \quad k = p - n$$

2.6 Hash Function

A hash function is a function that is used to map data of arbitrary size to a fixed size. The values returned by the hash functions are called a *hash* and the input data can be called the *key*. The main functions that hash functions performs are to convert variable-length keys into fixed length values, scramble the bits of the key so that the resulting hashes are uniformly distributed over the space or memory, then lastly, map the key values into ones smaller than the size of the table. A good hash function can satisfy two properties: efficient to compute and little collisions. Collisions are situations where two different data can produce the same hash. Fundamentally, mapping into a fixed size will use the equation:

$$h(K) = K \bmod m$$

Where m is the memory available, K is the key, and $h(K)$ will return the hash of the key.

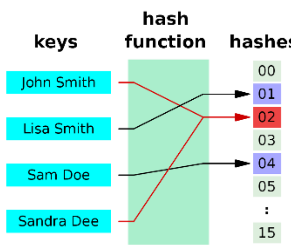


Fig. 2.6.1 Hash Function visualization
Source: https://en.wikipedia.org/wiki/Hash_function

2.7 Cryptography

Cryptography, in Greek, can be translated as "secret writing", as it is the practice and study of ways to secure communications of two or more side from an external side. Generally, cryptography can be described as configuring and analyzing protocols that prevent third parties from accessing the communication of two parties. The text to be secured, called the data or the plaintext, is to be encrypted, which is the act of making the plaintext into a ciphertext using an algorithm, and the ciphertext, is the data after it is encrypted, which has no meaning for a third party which has no access on the decryption algorithm, which is the act to convert the ciphertext back to plaintext. An early substitution cipher was the Caesar Cipher, which shifts the letter on the plaintext by a fixed number to produce the ciphertext.

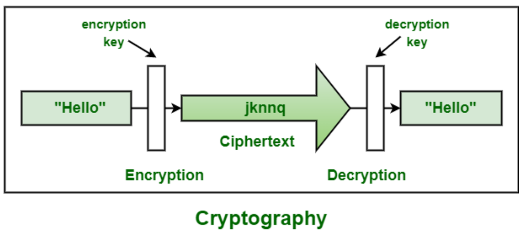


Fig. 2.7.1 Cryptography visualization
Source: https://www.geeksforgeeks.org/difference-between-encryption-and-cryptography/

2.8 RSA Algorithm

The RSA Algorithm, named after its founder, Ronald Rivest, Adi Shamir, and Leonard Adleman in 1976, is an asymmetric cryptography algorithm, that is, the encrypting key and decrypting key is separated from each other. The encrypting key is called the public key, as it is not kept secret from the public, same goes for the decrypting key, or the private key, it is only known to the key owner. The main procedure of the RSA Algorithm for key generation are as thus: Pick two secret prime numbers p and q, then calculate $n = pq$ and $m = (p-1)(q-1)$, where n need not be secret, but m has to, after that, pick a number e that is relatively prime to m, that is $GCD(e, m) = 1$, then we get the private key d from $ed \equiv 1 \ (mod\ m)$.

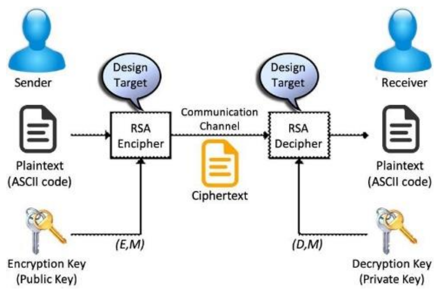


Fig. 2.8.1 RSA Algorithm visualization
Source: https://www.researchgate.net/figure/RSA-algorithm-structure_fig2_298298027

3. Cryptographic Hash Function

A cryptographic hash function is a mathematical algorithm that combines the use of hash functions and cryptography, where, by combining the two concepts, results in a one-way function that maps a data into a hash, but won't show the data from the hash. The only way to find the data that produces a specific hash is to attempt a brute-force search of possible inputs to see the matches. A cryptographic hash function has to be deterministic, where the same data will always produce the same hash if hashed, but will produce an entirely new hash if the data is changed even just a little bit, and like a good hash function, minimizes collisions. One of the most popular CHF algorithm used in modern technologies is the SHA-2 or Secure Hash Algorithm 2, which is a set of cryptographic hash functions designed by the USA National Security Agency. SHA-2 consists of two main algorithm, SHA-256 and SHA-512, where SHA-256 will map the value into a 256 bit (32 byte) value and the SHA-512 will map the value into a 512 bit (64 bytes) value.

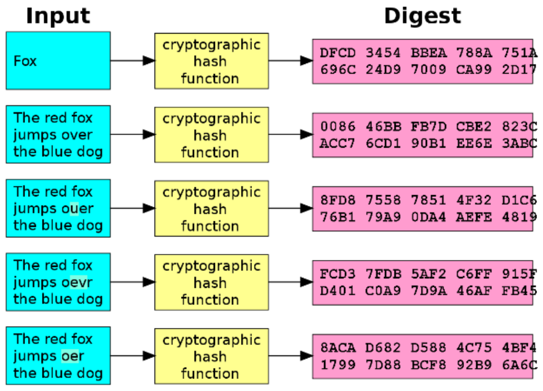

Input
Digest
Fox
The red fox jumps over the blue dog
The red fox jumps ouer the blue dog
The red fox jumps oevr the blue dog
The red fox jumps oer the blue dog
cryptographic hash function
DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17
0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC
8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819
FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45
8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C

Fig. 3.1 Cryptographic Hash Function visualization
Source: https://en.wikipedia.org/wiki/Cryptographic_hash_function

4. Blockchain

Blockchain can be defined as a type of Distributed Ledger Technology that consists of continuously growing blocks of data that is linked to one another by a CHF. Each block of data contains the hash of the previous block, a timestamp, the data stored, and when the data stored reaches the maximum size, a hash of the block (including the hash function of the previous block) will be generated, effectively generating a link from a block into another block, which in turn results in the safety of the previous block, as changing some value of the previous block, will also change its hash, which creeps into the next block, and so on. For security, the blockchain is typically managed by a peer-to-peer computer network, where nodes follow a consensus algorithm protocol to add a new block. This distribution principle makes it hard for one side to manipulate the data on the previous blocks, not like centralized data storage, thus the only way to manipulate data onto the block is to add it into the new block.

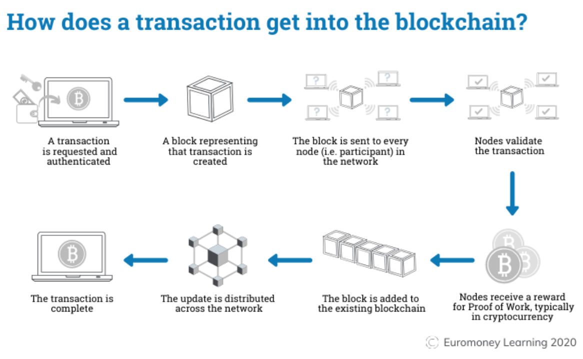

How does a transaction get into the blockchain?
A transaction is requested and authenticated
A block representing that transaction is created
The block is sent to every node (i.e. participant) in the network
Nodes validate the transaction
Nodes receive a reward for Proof of Work, typically in cryptocurrency
The block is added to the existing blockchain
The update is distributed across the network
The transaction is complete
© Euromoney Learning 2020

Fig. 4.1 How transactions/data gets into the blockchain
Source: https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain

III. Blockchain Implementation

A. Main Features

Blockchains, as implemented in the cryptocurrency such as *Bitcoin*, *Ethereum*, etc. has six main features:

- Immutability
- Decentralized
- Enhanced Security
- Distributed Ledgers
- Faster Settlements/Consensus

The main features that are to be implemented, and the main features for a blockchain data structure, is the immutability and enhanced security, the other four are network protocols, thus not making it the subject of the implementation. Immutability, as defined by the Merriam-Webster dictionary: "not capable of or susceptible to change.", meaning that the data stored in the blockchain, if it is already set, then the data cannot be changed anymore. Enhanced Security, in this sense, is the security of the data being public, yet it cannot be modified, and the only one that can make an input to how the data changes is the one holding on to the key to their own data that is in the blockchain. On the next part, it will be shown on how these two features can be implemented using graph theory and number theory.

B. Implementation

To implement the blockchain, first, it would be needed to separate it into different parts, the main parts of the blockchain are as such:

- Block – The main part that stores the data
- Link – Connects the block to one another and provides the possibility to implement a multi-branch checking for validity, where the protocol will prefer to use the longer chain as the valid chain, further improving the security of the blockchain
- Block Signature – The unique string that is generated from the block's data and previous block's signature by using a cryptographic hash function

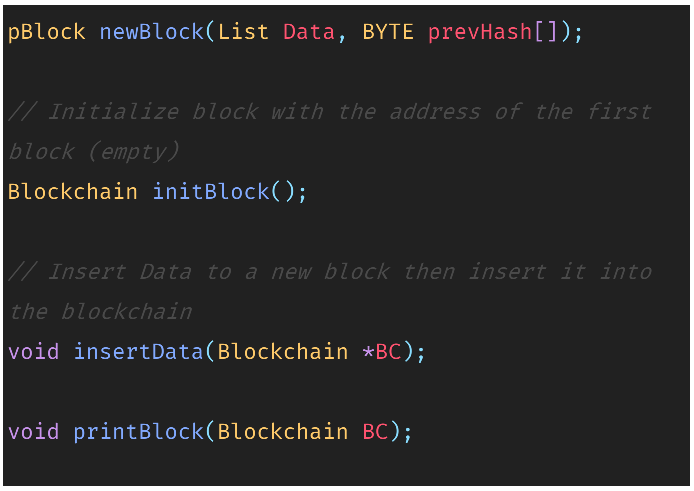
C. Implementation Product

The main features' implementation, is implemented using the C Language, using a Blockchain Data Type, the program structure is as follow:

- blockchain.c
- blockchain.h
- sha256_test.c
- sha256.c
- sha256.h
- test.c

The implementation for the blockchain datatype is in the blockchain.h file:

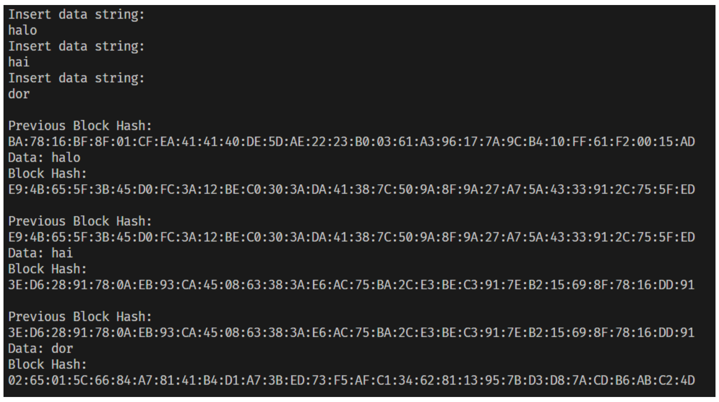
```c
typedef struct list
{
    /* data */
    BYTE buffer[256];
} List;

typedef struct block *pBlock;

typedef struct block
{
    /* data */
    BYTE prevHash[SHA256_BLOCK_SIZE];
    List data;
    BYTE blockHash[SHA256_BLOCK_SIZE];
    pBlock nextBlock;

} Block;

typedef struct blockchain
{
    pBlock first;
    pBlock last;
} Blockchain;

// SHA-256 Cryptographic Hash Function, Implemented by Brad Conte
void hash(List Data, BYTE *Hash[]);

void hashBlock(Block B, BYTE Hash[32]);

// Make a new block
```



```c
pBlock newBlock(List Data, BYTE prevHash[]);

// Initialize block with the address of the first
block (empty)
Blockchain initBlock();

// Insert Data to a new block then insert it into
the blockchain
void insertData(Blockchain *BC);

void printBlock(Blockchain BC);
```

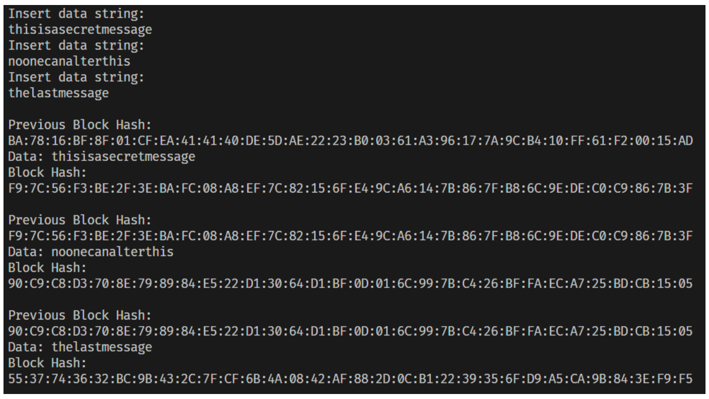
Fig 3.1 Blockchain datatype in C
Source: Personal

For the SHA-256 algorithm, the SHA-256 Cryptographic Hash Function implementation by Brad Conte will be used (Link: https://github.com/B-Con/crypto-algorithms). The result of the implementation is shown here:



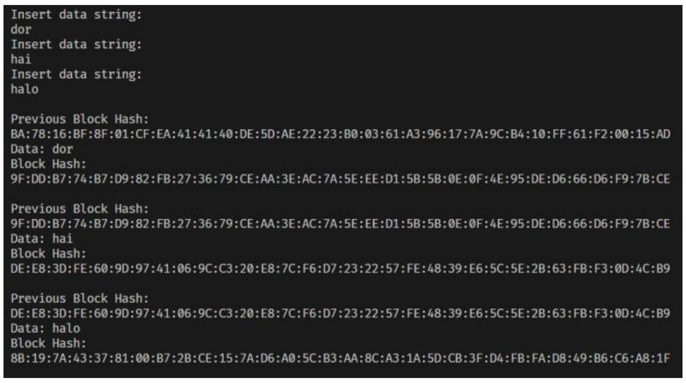
```
Insert data string:
halo
Insert data string:
hai
Insert data string:
dor

Previous Block Hash:
BA:78:16:BF:8F:01:CF:EA:41:41:40:DE:5D:AE:22:23:B0:03:61:A3:96:17:7A:9C:B4:10:FF:61:F2:00:15:AD
Data: halo
Block Hash:
E9:4B:65:5F:3B:45:D0:FC:3A:12:BE:C0:30:3A:DA:41:38:7C:50:9A:8F:9A:27:A7:5A:43:33:91:2C:75:5F:ED

Previous Block Hash:
E9:4B:65:5F:3B:45:D0:FC:3A:12:BE:C0:30:3A:DA:41:38:7C:50:9A:8F:9A:27:A7:5A:43:33:91:2C:75:5F:ED
Data: hai
Block Hash:
3E:D6:28:91:78:0A:EB:93:CA:45:08:63:38:3A:E6:AC:75:BA:2C:E3:BE:C3:91:7E:B2:15:69:8F:78:16:DD:91

Previous Block Hash:
3E:D6:28:91:78:0A:EB:93:CA:45:08:63:38:3A:E6:AC:75:BA:2C:E3:BE:C3:91:7E:B2:15:69:8F:78:16:DD:91
Data: dor
Block Hash:
02:65:01:5C:66:84:A7:81:41:B4:D1:A7:3B:ED:73:F5:AF:C1:34:62:81:13:95:7B:D3:D8:7A:CD:B6:AB:C2:4D
```

Fig. 3.2 Blockchain implementation using C
Source: Personal



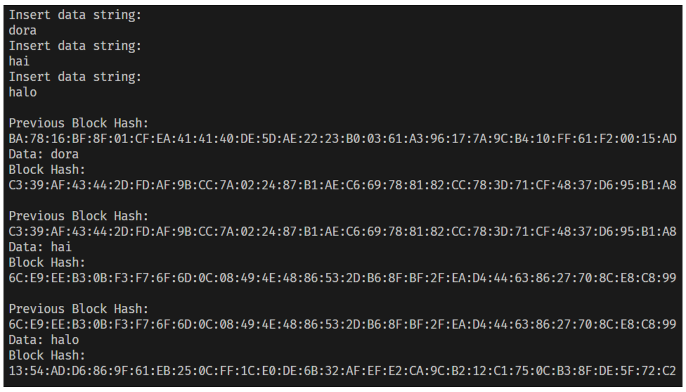
```
Insert data string:
thisisasecretmessage
Insert data string:
noonecanalterthis
Insert data string:
thelastmessage

Previous Block Hash:
BA:78:16:BF:8F:01:CF:EA:41:41:40:DE:5D:AE:22:23:B0:03:61:A3:96:17:7A:9C:B4:10:FF:61:F2:00:15:AD
Data: thisisasecretmessage
Block Hash:
F9:7C:56:F3:BE:2F:3E:BA:FC:08:A8:EF:7C:82:15:6F:E4:9C:A6:14:7B:86:7F:B8:6C:9E:DE:C0:C9:86:7B:3F

Previous Block Hash:
F9:7C:56:F3:BE:2F:3E:BA:FC:08:A8:EF:7C:82:15:6F:E4:9C:A6:14:7B:86:7F:B8:6C:9E:DE:C0:C9:86:7B:3F
Data: noonecanalterthis
Block Hash:
90:C9:C8:D3:70:8E:79:89:84:E5:22:D1:30:64:D1:BF:0D:01:6C:99:7B:C4:26:BF:FA:EC:A7:25:BD:CB:15:05

Previous Block Hash:
90:C9:C8:D3:70:8E:79:89:84:E5:22:D1:30:64:D1:BF:0D:01:6C:99:7B:C4:26:BF:FA:EC:A7:25:BD:CB:15:05
Data: thelastmessage
Block Hash:
55:37:74:36:32:BC:9B:43:2C:7F:CF:6B:4A:08:42:AF:88:2D:0C:B1:22:39:35:6F:D9:A5:CA:9B:84:3E:F9:F5
```

Fig. 3.3 Blockchain implementation using C, different data
Source: Personal



```
Insert data string:
dor
Insert data string:
hai
Insert data string:
halo

Previous Block Hash:
BA:78:16:BF:8F:01:CF:EA:41:41:40:DE:5D:AE:22:23:B0:03:61:A3:96:17:7A:9C:B4:10:FF:61:F2:00:15:AD
Data: dor
Block Hash:
9F:DD:B7:74:B7:D9:82:FB:27:36:79:CE:AA:3E:AC:7A:5E:EE:D1:5B:5B:0E:0F:4E:95:DE:D6:66:D6:F9:7B:CE

Previous Block Hash:
9F:DD:B7:74:B7:D9:82:FB:27:36:79:CE:AA:3E:AC:7A:5E:EE:D1:5B:5B:0E:0F:4E:95:DE:D6:66:D6:F9:7B:CE
Data: hai
Block Hash:
DE:E8:3D:FE:60:9D:97:41:06:9C:C3:20:E8:7C:F6:D7:23:22:57:FE:48:39:E6:5C:5E:2B:63:FB:F3:0D:4C:B9

Previous Block Hash:
DE:E8:3D:FE:60:9D:97:41:06:9C:C3:20:E8:7C:F6:D7:23:22:57:FE:48:39:E6:5C:5E:2B:63:FB:F3:0D:4C:B9
Data: halo
Block Hash:
8B:19:7A:43:37:81:00:B7:2B:CE:15:7A:D6:A0:5C:B3:AA:8C:A3:1A:5D:CB:3F:D4:FB:FA:D8:49:B6:C6:A8:1F
```

Fig. 3.4 Blockchain Implementation using C, different data orientation
Source: Personal



```
Insert data string:
dora
Insert data string:
hai
Insert data string:
halo

Previous Block Hash:
BA:78:16:BF:8F:01:CF:EA:41:41:40:DE:5D:AE:22:23:B0:03:61:A3:96:17:7A:9C:B4:10:FF:61:F2:00:15:AD
Data: dora
Block Hash:
C3:39:AF:43:44:2D:FD:AF:9B:CC:7A:02:24:87:B1:AE:C6:69:78:81:82:CC:78:3D:71:CF:48:37:D6:95:B1:A8

Previous Block Hash:
C3:39:AF:43:44:2D:FD:AF:9B:CC:7A:02:24:87:B1:AE:C6:69:78:81:82:CC:78:3D:71:CF:48:37:D6:95:B1:A8
Data: hai
Block Hash:
6C:E9:EE:B3:0B:F3:F7:6F:6D:0C:08:49:4E:48:86:53:2D:B6:8F:BF:2F:EA:D4:44:63:86:27:70:8C:E8:C8:99

Previous Block Hash:
6C:E9:EE:B3:0B:F3:F7:6F:6D:0C:08:49:4E:48:86:53:2D:B6:8F:BF:2F:EA:D4:44:63:86:27:70:8C:E8:C8:99
Data: halo
Block Hash:
13:54:AD:D6:86:9F:61:EB:25:0C:FF:1C:E0:DE:6B:32:AF:EF:E2:CA:9C:B2:12:C1:75:0C:B3:8F:DE:5F:72:C2
```

Fig. 3.5 A slight change in the initial block will result in a completely different hash for its successor blocks
Source: Personal

The full implementation repository can be viewed here. Note that when the same data is fed into the program, the same hash will appear, as it is the nature of the Cryptographic Hash Function, but if the data's orientation is switched, then it will result in a different hash as the hash of the previous data is also included in the block's hash.

D. Review

The Blockchain, a Data Type which guarantees a secure storage of data, is essentially a graph with a combination of number theory for its relations. Where the vertices will store the data, and the edges connects one vertex to another, this edge is in a way secured, because the edge cannot be broken once made, because of the cryptographic hash function that acts on the vertices. For debate, the implemented version of the blockchain is a simplification model of the real blockchain, as it only implements a single, continuous chain, so, in a way, it can be categorized as a tree, as trees are a subset of graphs. But real blockchains have a multi-chain principle for its consensus to add blocks to the chain, as the main chain will choose the longest branch, thus making it, in a way, a graph.

The number theory application in blockchain is also fundamental, as a cryptographic hash function combines cryptography and hash functions, which is the applications of number theory.

IV. Review

Blockchain is essentially a modification of a graph datatype, modified with the concepts of number theory. Thus, making it a perfect, really useful example of why graph theory and number theory is really important, especially in Computer Science and Security. As blockchains are heavily used, and will continue to improve its usage, it will be important that the concepts that fundamentally supports it gets the acknowledgement it deserves. In this paper, the model of a blockchain is implemented using the C language and the knowledge of Abstract Data Type, but it can be also implemented using Object-Oriented Programming. The implemented version of the blockchain in this paper is only a model of the real blockchain, but it covers the main aspect of the blockchain, its reliability, its data security, the chain of hashes, its cryptographic hash functions, and the link of blocks.

V. ACKNOWLEDGMENT

The Author would like to thank, first, the lecturer of Class 1 of Discrete Mathematics, Mrs. Nur Ulfa Maulidevi of Bandung Institute of Technology, as the materials given are presented in a way that can be fully understood deeply by The Author. Next, The Author would also like to thank Mr. Rinaldi Munir for assigning this paper, as it is a way of exploring the applications of the materials given in class. Lastly, The Author would also like to thank Mrs. Yani Widyani, as the lecturer of Class 1 Algorithm and Data Structure, as this application of the blockchain is only possible because of the Lectures given in the subject.

REFERENCES

[1] https://www.ibm.com/id-en/topics/data-security , accessed 3 December 2022, 12.57 P.M.
[2] https://blog.boot.dev/cryptography/how-sha-2-works-step-by-step-sha-256/ , accessed 3 December 2022, 13.15 P.M.
[3] https://www.investopedia.com/terms/p/proof-work.asp , accessed 3 December 2022, 13.28 P.M.
[4] https://www.mycryptopedia.com/sha-256-related-bitcoin/ , accessed 10 December 2022, 13.47 P.M.
[5] https://cryptocurrencyworks.com/.res/doc/Bitcoin-SNakamoto-Oct-2008.pdf , accessed 10 December 2022, 13.50 P.M.
[6] http://ijarse.com/images/fullpdf/1483098559_N218ijarse.pdf , accessed 10 December 2022, 14.27 P.M.
[7] https://github.com/B-Con/crypto-algorithms , accessed 11 December 01.40 A.M.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Desember 2022

[illegible]

Kenneth Ezekiel Suprantoni, 13521089