Journal of Science and Technology Research

Journal homepage: www.nipesjournals.org.ng

Comparison of Data Structure Techniques used for the Implementation of Digital Currency Payment Models

Aigbe Princewill[a], *Nwelih Emmanuel*[b]
[a] Department of Mathematics & Computer Science Western Delta University Oghara, Nigeria
[b]Department of Computer Science University of Benin, Benin City, Nigeria
Email: agbonx@yahoo.com, emmanuel.nwelih@uniben.edu

Article Info

***Keywords*:**
Data structure, digital currency, payment, electronic currency

Received 01July 2022
Revised 12 July 2022
Accepted 13 July 2022
Available online 2 Sept 2022

https://doi.org/10.37933/nipes/4.3.2022.4

https://nipesjournals.org.ng

Abstract

The emergence of electronic business has generated new payment necessities that in many circumstances cannot be feasibly fulfilled by the fiat payment systems. The success of electronic commerce business depends on the credibility of the available electronic payment systems. Digital currency payment models were designed to settle payment transactions online, but experiences the issue of double-spending fraud. Double-spending refers to the inability of digital currency payment systems permitting the potentiality to spend the same units of digital currency identification more than once. A number of models to combat double-spending fraud have been designed and developed. Some of the existing schemes prevent double-spending fraud after digital currency payment transactions have been completed. Therefore, this paper focuses on the determination of the digital currency payment model that has the best computational time to prevent double-spending fraud in payment transactions. This is done by the procedural study of the data structure techniques used by the digital currency payment models that prevent double-spending fraud before it occurs in terms of data representation, implementation and method used to prevent double-spending fraud and the required search time. The appraisal culminated in the derivation of the time complexity of each of the data structure techniques used for the implementation of the digital currency payment models.

1. Introduction

The exchange of goods and services conducted face-to-face between two parties' dates back to before the beginning of recorded history. Eventually, as trade became more complicated and inconvenient, humans invented abstract representations of value. As time passed, representations of value became more and more abstract, progressing from barter through bank notes, payment orders, cheques, credit cards, and now electronic payment systems. Traditional means of payment suffer from various well-known defects or problems; Money can be counterfeited, signatures forged, and cheques bounced [1]. On the other hand, properly designed electronic payment systems can actually provide better security than traditional means of payments, in addition to flexibility of use [2].

E-commerce provides the capability of buying and selling goods, services and information on the internet by using electronic payment systems. In electronic payment systems, the exchange of value is represented by the exchange of data and it is easy, cheap and fast to transfer data [3]. Securely transfer monetary value can establish electronic business participant confidence and facilitate the electronic transaction process [4]. Electronic payment systems mechanisms are the means of payment for online purchases. An e-commerce payment system facilitates the acceptance of electronic payment for online transactions. Also known as a sample of Electronic Data Interchange (EDI), e-commerce payment systems have become increasingly popular due to the widespread use of the internet-based shopping and banking [5]. As the use of the different electronic payment methods has become more widespread, so is the amount of fraud related to these payment mechanisms [6]. Fraudsters are becoming more organized and are using increasingly sophisticated methods (such as phishing, account takeover, counterfeiting, stolen payment instrument, transaction repudiation, etc) to obtain and misuse consumer personal and financial information. Electronic payment fraud is causing billions of dollars in losses for the electronic commerce industry [7]. Besides direct losses, the brand name can be affected by loss of consumer confidence due to the fraud [8]. As a result of these growing losses, financial institutions and other electronic payment industry stakeholders are continually seeking new techniques and innovations in fighting electronic payment fraud [9].

2.0 Overview of Digital Currency Payment System

Over the years, different electronic payment methods have emerged for settlement of payments for e-commerce transactions. These payment mechanisms include electronic cash payment (eCash), electronic cheque (eCheque), online credit card, and smart card. The online merchants have to comply with stringent rules stipulated by the available electronic payments they support. This means that merchants must have security protocol and procedures in place to ensure transactions are more secure [10]. This can also include having a certificate from an authorized certification authority (CA) who provides Public-Key infrastructure (PKI) for securing online payment transactions [11].

There are intermediaries that enable financial transactions to take place over the internet. Many of the intermediaries permit consumers to establish an account quickly, and to transfer funds into their on-line accounts from a traditional bank account and vice versa, after verification of the consumer's identity and authority to access such bank accounts. Also, the larger intermediaries further allow transactions to and from credit card accounts, although such credit card transactions are usually assessed a fee (either to the recipient or the sender) to recoup the transaction fees charged to the intermediary. The speed and simplicity with which cyber-intermediary accounts can be established and used have contributed to their widespread use [12].

The first untraceable digital currency (eCash) payment system based on blind signature was proposed by [13], which allows the requester to obtain a message signature from a signer without revealing the message content and makes the signer unable to link any signed message to its signature. This initial proposal required an online broker to clear eCash before merchants could provide their services. To protect against double-spending. This means that eCash or digital currency customers or users presented for payment to settle a given transaction must be verified by online electronic cash brokers. The implication here is that there will be many online electronic cash brokers as users or customers will likely choose their online brokers, making the eCash payment system complicated and very expensive.

Furthermore, the digital currency payment system introduced to settle transactions electronically was associated with a payment fraud known as double-spending. Double-spending is the failure of a digital currency payment system allowing the possibility to spend the same or single digital currency (eCash) more than once. Different digital currency payment schemes or algorithms have been proposed with various security features to prevent or fight against double-spending. The different techniques that have been used by digital currency payment to fight against double-spending resulted in double-spending fraud occurring before its detection. A trace of the payment transaction is carried out to determine the double-spender [14]. However, the application of some specific data structure techniques with salient features such as block chain, binary tree, and hash table in digital currency payment systems have successfully prevented double-spending fraud in digital payment system transactions before occurring. These data structure techniques can be used to store and search for a given unit of digital currency identification in payment transactions with effective time complexities [15].

3.0 Digital Currency Payment Transaction using Blockchain Data Structure0

Structurally, a blockchain can be defined as a linked list (blocks) of a group of transactions which is connected with each other using hash pointers rather than pointers as in the case of the linked list. The blockchain in itself is a data structure that stores transactions. It is similar to a linked list in that the data is split into containers known as blocks. Each block is connected with its predecessor with a cryptographically secured reference [16]. The secured references establish order throughout the blocks and effectively make the blockchain an *apend-only* data structure where new data can only be added with new blocks. The hash pointer is not only used to look up the previous block of a transaction but also used to verify that the transactions stored in the previous block is not tampered. The hash pointer is the hash value of the header data of the previous block also known as block header. The hash value of the previous block header is included in the following block as a reference because the *block hash* depends on the data of a block, even changing a single character in one of the transactions would invalidate the reference. A detailed components of blockchain and a set of transactions is shown in a block diagram in Figure 1.

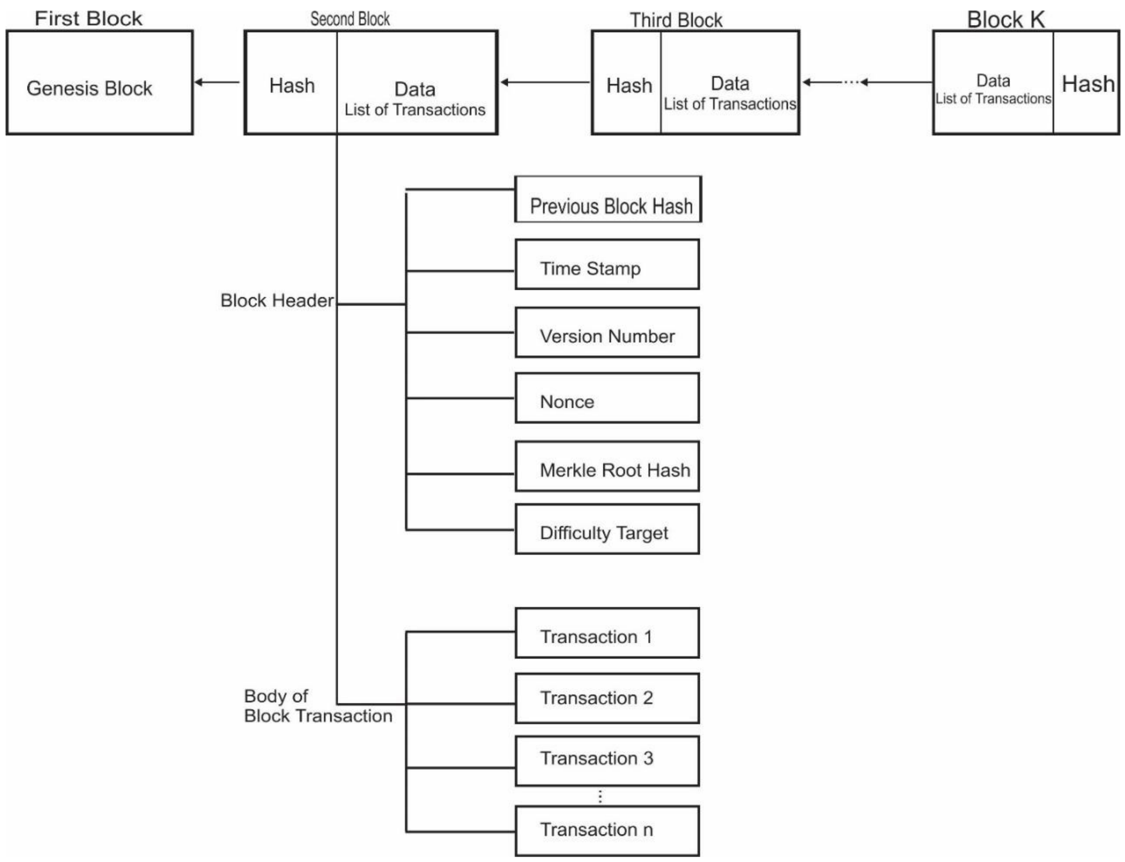


Figure 1: Components of a blockchain

Blockchains are often used in cryptocurrencies because they can function as public ledgers. Each block stores a batch of financial transactions, and blocks are added to the blockchain through a consensus algorithm. This combination of a blockchain and a consensus algorithm for adding blocks to the chain allows for decentralized digital currencies. Cryptocurrency holders cannot double spend their digital money since transactions are recorded in a tamper-resistant public ledger [17]. Each block consists of a header section and a body section. The header section of the block may consist of the following information:

i. Merkel root
The Merkel root represents a summary of all transactions included in the block.

ii. Nonce
The number used once (Nonce) is the variable that miners change to modify the block headers hash for its value to meet the specific difficulty.

iii. Timestamp
The Time is the time when the miner started hashing the header for the mining process.

iv. Hash of the previous block header
The previous block hash header serves two purposes. First, it establishes an order throughout the chain of blocks, and second, it ensures no preceding block can be changed without affecting the current and all subsequent blocks.

v. Version number
The Version number indicates which software version the miner of the block used and which set of block validation rules were followed.

vi. Difficulty target
The b*its* (or nBits*)* are an encoded version of the current difficulty of finding a new block.

The body section of the block may consist of the list of transaction data. When the blockchain data structure is used to represent digital currency, a block which is a unit of information is about 80 bytes of storage distributed as follows:

i. Merkel root (32 bytes)
ii. Hash of previous block header (32 bytes)
iii. Timestamp (4 bytes)
iv. Version number (4 bytes)
v. Nonce (4 bytes)
vi. Difficulty target (4 bytes)

A thorough search operation with confirmation mechanism through the entire blockchain is required to prevent double-spending fraud in payment transactions when a unit of digital currency value with a specific identification (DCid) is represented by a customer to settle transaction with a merchant [18].

Suppose a customer has 1 unit of digital currency identification ($DCid_1$) and try to spend it twice. The customer made 1 unit of digital currency (DC_{id1}) transaction (T1) to merchant1. Again, the customer signs and sends the same 1 unit of digital currency ($DCid_1$) transaction (T2) to merchant2. Both transactions go into the pool of unconfirmed transactions where many unconfirmed transactions are stored already. The unconfirmed transactions are transactions which do not pick by anyone. Now, whichever transaction first got confirmations and was verified by miners, will be valid. Another transaction which could not get enough confirmations will be pulled out from the network. A block diagram of the illustration of both transactions is shown in Figure 2. Transaction T1 is valid, and merchant1 will receive the 1 unit of digital currency ($DCid_1$).

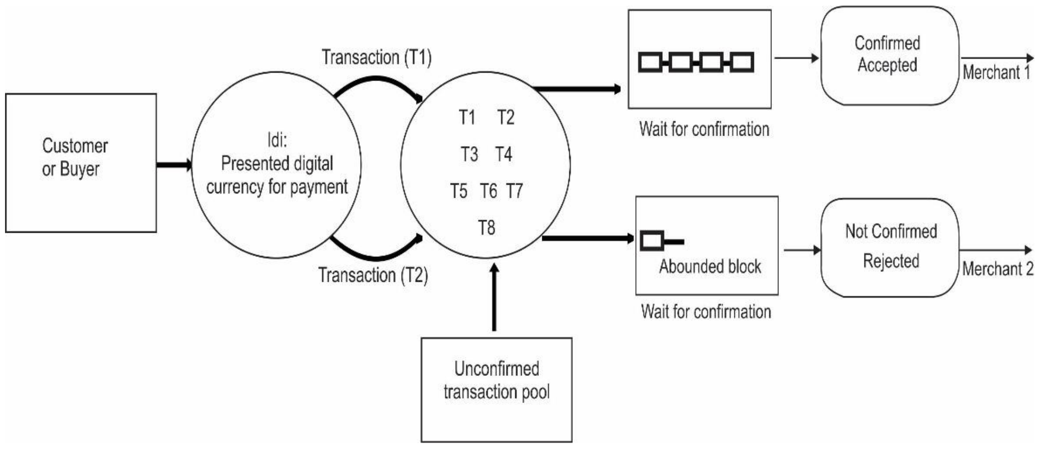


Figure 2: Prevention of double-spending fraud in a blockchain implemented digital currency payment model

The confirmation is achieved after an elaborate search for the presented unit of digital currency (id_1) for payment transaction in all the blocks of the chain to ensure that it has not be spent. This is possible as the summary of the transactions in the blocks are stored by Merkle trees [19]. A search for the unit of digital currency in the Merkle tree yields a time complexity of order $\log_2 n$), where n is the number of units of digital currency in the blockchain. If there are k blocks in the chain, all the k blocks need to be searched. The search time t(n) for a given unit of digital currency (id_1) in a blockchain data structure during a payment transaction is:

$$t(n) = k * (\log_2 n) \quad (2)$$

Since there are k blocks in the blockchain therefore, the complete search time t(n) is derived as:

$$t(n) = O(k * (\log_2 n)) \quad (2)$$

Thus, on an average searching has $O(k * (\log_2 n))$ in a blockchain. This means that given an input size n (that is, digital currency identification keys), the number of key comparisons (search time) required to prevent double-spending fraud before it occurs is $O(k * (\log_2 n))$ when a blockchain data structure is used.

4.0 Digital Currency Payment Transaction using Binary Tree Data Structure

A digital currency payment model that prevents double-spending fraud before it occurs was designed [20]. This model was implemented using the binary tree data structure in which the nodes of the tree represent units of digital currency identifications (DCids). In the binary tree, each divisible unit of digital currency identification (DCid) of a given monetary value 2^L, is assigned to a binary tree of L +1 levels. The values of the leaves are the least, namely 1. Each of the leaves node is assigned a key denoted by $DCid_{L+1}$, j, where $0 \leq j \leq 2^L -1$. Any other internal node corresponds to an amount of money which is exactly twice the amount of their corresponding child node values and also is assigned a key defined by $DCid_{i,j}$. The root node lies in the 0th level and has a maximum value, namely, 2^L. The corresponding key of the root node is $DCid_{0,0}$.

The binary tree is constructed from the leaves node. Each internal node is obtained by the multiplication of its' corresponding leaves node. When the user wants to spend a node, he supplies the keys of the spent node and the corresponding leaves node to the merchant. The user generates the internal node only by the multiplication of its corresponding nodes. The binary tree is given in Figure 3.

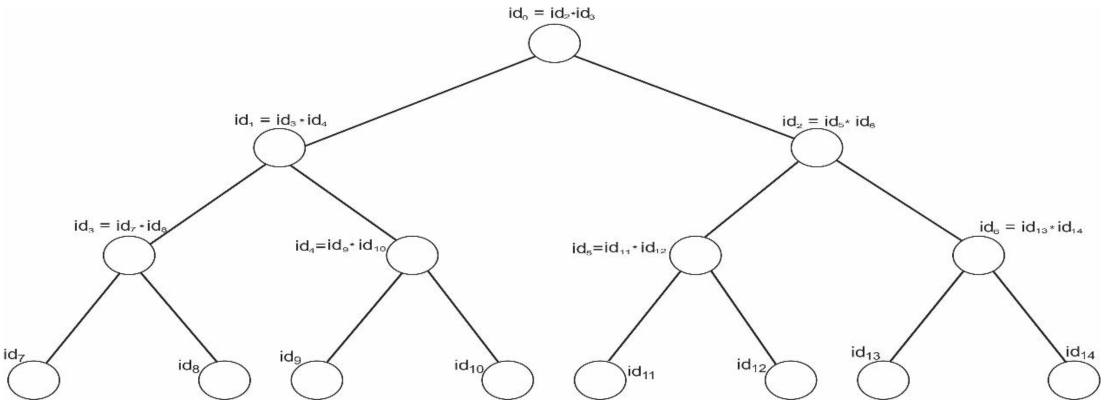


Figure 3: Construction of a binary tree (L = 3) of digital currency identifications

The [20] digital currency payment model determines double-spending and identity of double-spender in a payment process as follows:
The merchant deposits a coin $co_M = \{2^1, S, DCid, T, R, r1, \pi_s\}$ to bank, where $id = s_{j2}{}^1 || s_{j2}{}^1 + 1 || ... || s_{(j+1)2}{}^1 - {}_1$. At first, the bank checks the proof π_s . If it's not correct, the bank rejects the deposit. Otherwise, bank checks if these elements $s_{j2}{}^1, s_{j2}{}^1+1, s_{(j+1)2}{}^1{}_{-1}$ are already in the database. If one of these random elements is already in the database, bank executes the procedure of double-spender identification. Otherwise, bank adds 2^1 leaves node into the database. Then bank checks whether R is fresh. If R is fresh, bank accepts the coin $\{2^1, S, DCid, T, R, r1, \pi_s\}$, credits merchant's account. Otherwise, merchant deposits the coin twice. Bank refuses the deposit and warns merchant.

Bank obtains two coins $co_1 = \{2^1{}_1, S_1, DCid_1, T_1, R_1, r1_1, \pi_{S_1}\}$ and $co_2 = \{2^1{}_2, S_2, DCid_2, T_2, R_2, r1_2, \pi_{S_2}\}$. If the user spends the same node, then

$co_1 = \{2^1{}_1, S_1, DCid_1, T_1, R_1, r1_1, \pi_{S_1}\}$ and $co_1' = \{2^1{}_2, S_2, DCid_2, T_2, R_2, r1_2, \pi_{S_2}\}$. Thus, bank computes $pk_u = (T_1{}^R{}_2 / T_2{}^R{}_1)^{1/R2 - R1}$; if user spends the different nodes, then $co_1 = \{2^1{}_1, S_1, K_1, T_1, R_1, r1_1, \pi_{S_1}\}$ and $co_1' = \{2^1{}_2, S_2, DCid_2, T_2, R_2, r1_2, \pi_{S_2}\}$.

This digital currency payment scheme allows the user to withdraw a single divisible unit of digital currency identification and spends the sub-eCash by dividing the value of the electronic cash. The scheme protects against double-spending by conducting a look up of the unit of digital currency DCid number in a table of previously spent digital curency. This process requires key comparisons. The number or amount of comparisons determine the search time to locate a given digital currency id. Shown in Figure 4 is a binary search tree in which the nodes represent digital currency identifications or ($eCash_{id}$):

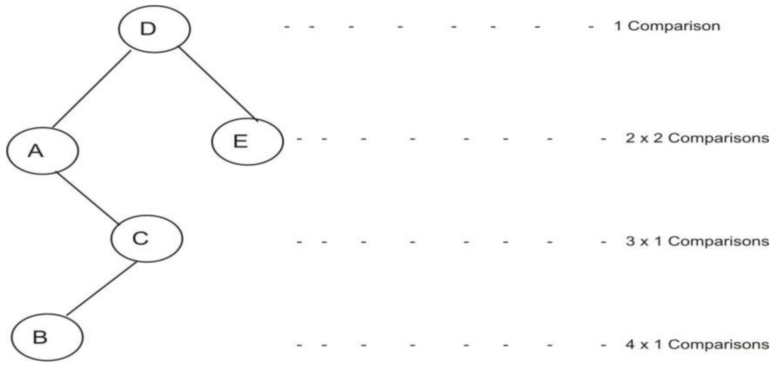


Figure 4: Key comparison in Binary tree

The number of comparisons required to search for given node in a binary tree depend on the depth of that node [21]. The number of elements (or nodes) in Figure 4 is 5. Therefore, average number of comparisons for a successful search in this binary tree in 12/5. Therefore, average number of comparisons increase if elements (nodes) are far away from root. For calculating the average number of comparisons required in the case of an unsuccessful search, we use the concept of extended Binary tree. An extended binary tree is a binary tree in which special nodes are added whenever a null sub-tree was present in the original tree so that each node in the original tree (except the root node) has degree three. A binary tree with dummy nodes is shown in Figure 5.

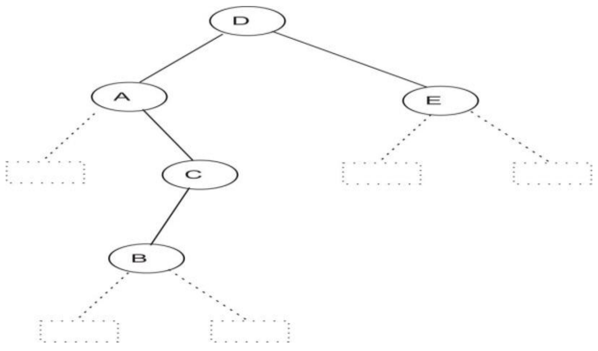


Figure 5: Binary tree with dummy nodes

The number of dummy nodes in a tree with n nodes is n + 1. By induction, let us assume a tree with n – 1 node. By our assumption, it has n dummy nodes. If we add 1 more node, I dummy node is deleted and 2 are added. Thus, the new number of dummy nodes in a tree n – 1 + 1 nodes = n – 1 + 2 = n + 1. Thus our assumption is valid.

Internal Path length:	I_n = sum of the path lengths of the tree internal nodes from the root in a tree with n nodes.
External Path Length:	E_n = Sum of the path lengths of the dummy nodes from the root in a tree with n nodes.

Hence the average number of comparisons for successful search is

$$S_n = \frac{I_n + n}{n} \quad (3)$$

and the average number of comparisons for an unsuccessful search is

$$U_n = \frac{En}{n+1} \quad (4)$$

The relation between I_n and E_n is given as: $E_n = I_n + 2_n$

By induction, consider a tree with n – 1 node. To construct T_n from T_{n-1}, we must replace a dummy node (whose path length from the dummy node is, say 1) with a leaf node plus two dummy nodes:

$$\begin{aligned} I_n &= I_{n-1} + l \\ E_n &= E_{n-1} - l + 2(l + 1) \\ &= E_{n-1} + l + 2 \\ &= I_{n-1} + 2n - 2 + I_n - I_{n-1} + 2 \\ &= I_n + 2_n \end{aligned}$$

Thus, $$\mathrm{Un} = \frac{\mathrm{I} + 2\mathrm{n}}{\mathrm{n} + 1} \quad (5)$$

Eliminating I from (3) and (5), we have

$nS_n = n + (n + 1) U_n – 2n$

$$Sn = \left(1 + \frac{1}{n}\right) U_n - 1 \quad (6)$$

Let’s now think about the order in which the elements are inserted in a binary tree as shown in Figure 6.

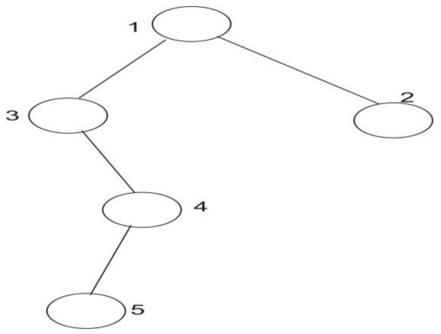


Figure 6: Order of elements insertion in a binary tree

Average number of comparisons for unsuccessful search at insertion number 2 = U_1. Thus if the key is there, for successful search at insertion number 2, number of comparisons required equals $U_1 + 1$ as shown in Figure 7.

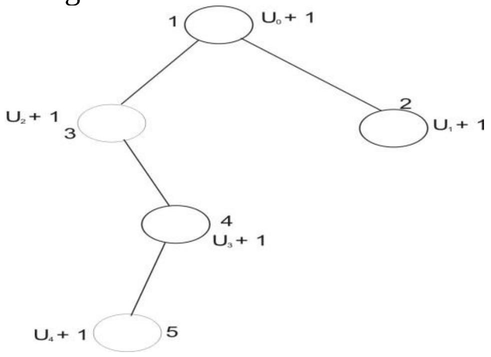


Figure 7: Searching in binary tree

The average number of comparisons corresponding to the “ith” inserted node for successful search = $U_{i-1} + 1$.

$$S_n = \frac{U_O + 1 + U_1 + 1 + \cdots + U_{n-1} + 1}{n} \tag{7}$$

Equating (6) and (7)

$$\frac{(n + 1)\, Un - n}{n} = U_O + 1 + U_1 + \cdots U_{n-1} + n$$

$$(n + 1)\, U_n = U_O + U_1 + \ldots + U_{n-1} + 2n \tag{8}$$

Substituting n by n – 1 we have,

$$nU_{n-1}U_O + U_1 + \ldots + U_{n-2} + 2\,(n-1) \tag{9}$$

Subtracting (8) from (7) we have,

$$(n + 1)\, Un - nU_{n-1} = U_{n-1} + 2$$

$$U_n = U_{n-1} + \frac{2}{n+1}$$

Now,

$$U_o = 0,\ U_1 = 1,\ U_2 = 1 + \frac{2}{3},\ U_3 = 1 + \frac{2}{3} + \frac{2}{4}$$

$$U_n = 1 + \frac{2}{3} + \frac{2}{4} + \ldots + \frac{2}{n+1} = 2(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{2}{n+1}) - 2$$

$$\approx 2l_n(n) - 2 = \log_2 n - 2$$

$$= O(\log_2 n). \tag{10}$$

Thus, on an average searching has $O(\log_2 n)$ in a binary tree. This means that given an input size n (that is, digital currency identification keys), the number of key comparisons (search time) required to prevent double-spending fraud before it occurs is $O(\log_2 n)$ when a binary search tree data structure is used.

5.0 Digital Currency Payment Transaction using Hash Table Data Structure

Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data [21]. A hash function (or hash code) is used by hash table to compute an index into buckets or slots from which the desired value can be found.

The digital currency identifications ($DCid_{ks'}$) generated by hash-cash algorithm are stored in hash table as shown in Figure 7. Where $DCidk_s$ are the generated identification keys for the digital currency values.

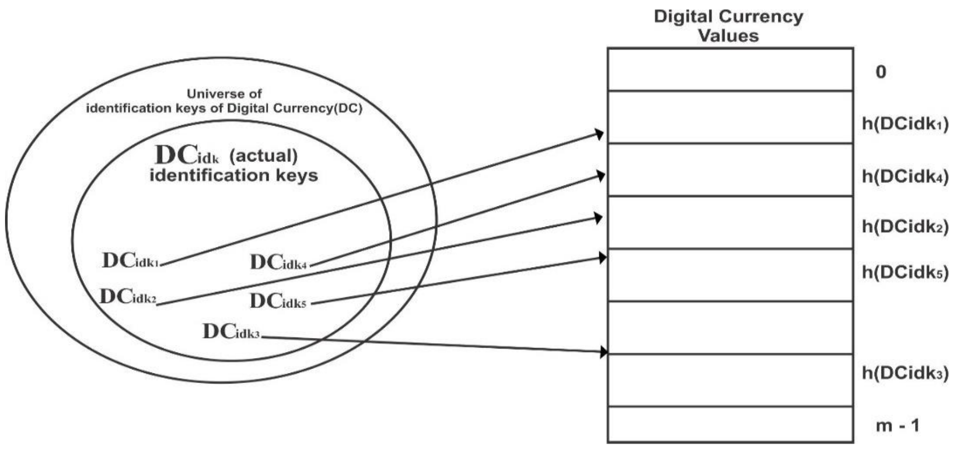


Figure 7: Hash table digital currency values and identifications mapping

In using the hash table data structure technique to prevent double spending before it occurs, every digital currency identification generated ($DCid_k$) has a control number known as the secret code [22]. This secret code is used to track or determine if a given digital currency unit identification ($DCid_k$) has been spent or not. Figures 8a and 8b show illustrate the necessary steps to prevent double-spending fraud.

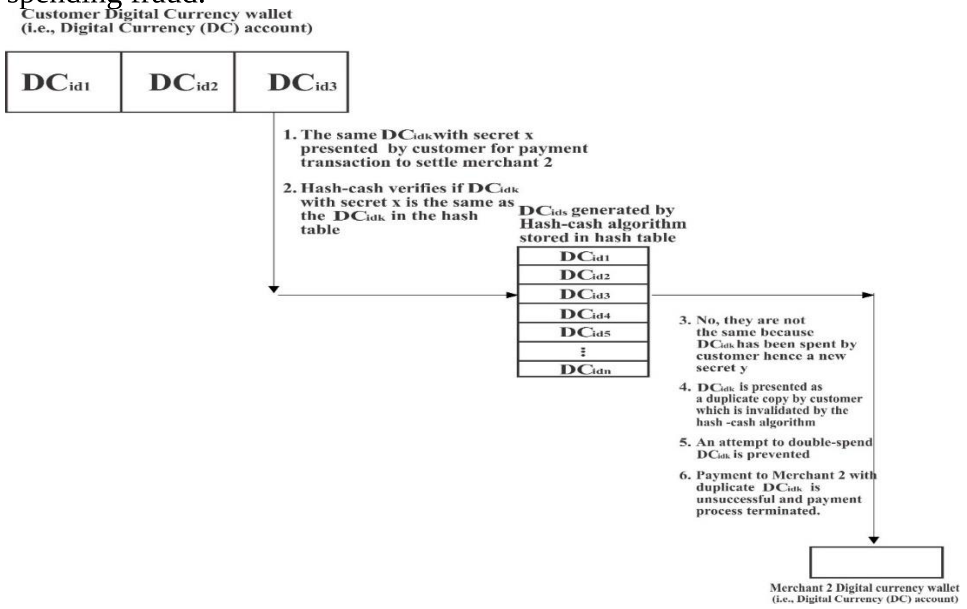
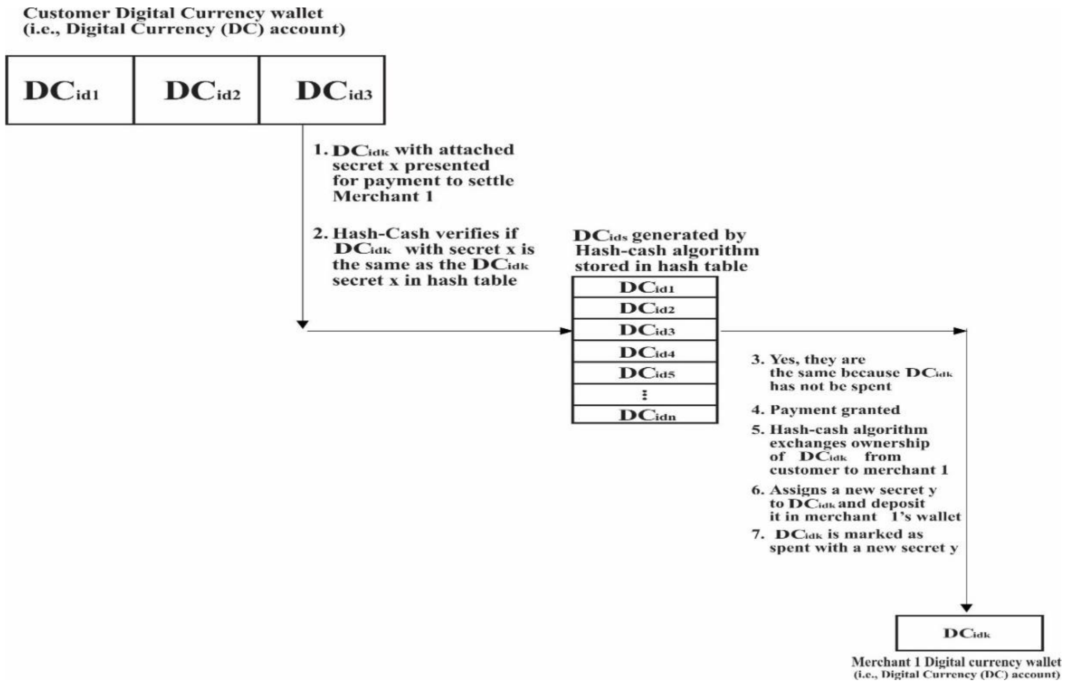

Customer Digital Currency wallet (i.e., Digital Currency (DC) account)
DC_{id1}
DC_{id2}
DC_{id3}
1. The same DC_{idk} with secret x presented by customer for payment transaction to settle merchant 2
2. Hash-cash verifies if DC_{idk} with secret x is the same as the DC_{idk} in the hash table
DC_{ids} generated by Hash-cash algorithm stored in hash table
DC_{id1}
DC_{id2}
DC_{id3}
DC_{id4}
DC_{id5}
⋮
DC_{idn}
3. No, they are not the same because DC_{idk} has been spent by customer hence a new secret y
4. DC_{idk} is presented as a duplicate copy by customer which is invalidated by the hash -cash algorithm
5. An attempt to double-spend DC_{idk} is prevented
6. Payment to Merchant 2 with duplicate DC_{idk} is unsuccessful and payment process terminated.
Merchant 2 Digital currency wallet (i.e., Digital Currency (DC) account)

Figure 8a: Prevention of double-spending fraud in a payment transaction


Customer Digital Currency wallet (i.e., Digital Currency (DC) account)
DCid1
DCid2
DCid3
1. DCidk with attached secret x presented for payment to settle Merchant 1
2. Hash-Cash verifies if DCidk with secret x is the same as the DCidk secret x in hash table
DCids generated by Hash-cash algorithm stored in hash table
DCid1
DCid2
DCid3
DCid4
DCid5
⋮
DCidn
3. Yes, they are the same because DCidk has not be spent
4. Payment granted
5. Hash-cash algorithm exchanges ownership of DCidk from customer to merchant 1
6. Assigns a new secret y to DCidk and deposit it in merchant 1's wallet
7. DCidk is marked as spent with a new secret y
DCidk
Merchant 1 Digital currency wallet (i.e., Digital Currency (DC) account)
Figure 8b: Prevention of double-spending fraud in a payment transaction

In hash table mapping operation, the load factor α which is the average key per slot is given as α = n/m (11)
Where m is the number of slots,
n is the number of elements (that is, number of eCash identification keys) stored in the hash table.
Let eCash identification key be element x_i, and x_i be the ith element inserted into the table, and let k_i = key[x_i] (12)
In a payment transaction involving eCash identification element x_i, define indicator random variable
$X_{ij} = I\{h(k_i) = h(k_j)\}$, for all i, j (13)
In simple uniform hashing, any key is equally likely to hash into any of the m slots independent of where any other hashes to. Then the probability P is given as

$$P\{h(k_i) = h(k_j)\} = 1/m \text{ and } E[X_{ij}] = 1/m. \quad (14)$$

Expected number of elements examined in a successful search is:

$$= E\left[\frac{1}{n}\sum_{i=1}^{n}\left(1 + \sum_{j=i+1}^{n} X_{ij}\right)\right] \quad (15)$$

Opening up brackets and taking expectation, we have

$$= \frac{1}{n}\sum_{i=1}^{n}\left(1 + \sum_{j=i+1}^{n} E[X_{ij}]\right) \quad (16)$$

From Equation 14, $E[X_{ij}] = 1/m$ and substituting this in Equation (16), we have,

$$= \frac{1}{n}\sum_{i=1}^{n}\left(1 + \sum_{j=i+1}^{n}\frac{1}{m}\right) \quad (17)$$

Since $\sum_{i=1}^{n} 1 = n$ and $\sum_{i=1}^{n}\sum_{j=i+1}^{n} = \sum_{i=1}^{n}(1 + (n-i))$ we have,

$$= 1 + \frac{1}{nm}\sum_{i=1}^{n} 1 + (n-i) \quad (18)$$

$$= 1 + \frac{1}{nm}\left(\sum_{i+1}^{n} n - \sum_{i=1}^{n} i\right) \quad (19)$$

Also $\sum_{i=1}^{n} = n$ and $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ and substituting these values into Equation 19, we have

$$= 1 + \frac{1}{nm}\left(n^2 - \frac{n(n+1)}{2}\right) \quad (20)$$

Simplifying we have,

$$= 1 + \left(\frac{n2}{nm} - \frac{1}{nm}\left(\frac{n(n+1)}{2}\right)\right)$$

$$= 1 + (\frac{n}{m} - \frac{(n+1)}{2})$$
$$= 1 + \frac{2n - n + 1}{2m}$$
$$= 1 + \frac{n-1}{2m} \qquad (21)$$

From Equation (21), α = $^{n}/_{m}$ which implies m = $^{n}/_{\alpha}$ and substituting into equation 20, we have

$$= 1 + \frac{\propto}{2} - \frac{\propto}{2n}$$

The expected total time for a successful search is

= Time to compute hash function + Time to search

$= O(1 + \frac{\propto}{2} - \frac{\propto}{2n}) = O(1+\propto)$

If n = O(m), then $\propto$ = n/m

= O(m)/m

= O(1). (22)

Thus, on an average searching has O(1). This means that given an input size n (that is, digital currency id_{ks}) in hash table data structure, the number of key comparisons (search time) required to prevent double-spending is O(1).

6.0 Search Time Performance Comparison of the Three Data Structures: Blockchain, Binary Tree, and Hash Table

The average case analyses of the three data structure techniques used for the implementation of digital currency payment transactions show that:
Blockchain data structure used by [14] digital currency payment model requires O(k * ($\log_2$ n)) number of key comparisons to prevent double-spending fraud before it occurs in a digital payment transaction. Binary tree data structure used by [19] digital currency payment model requires O($\log_2$ n) number of key comparisons to prevent double-spending fraud before it occurs in a digital currency payment transaction. Hash table data structure used by [23] digital currency payment model requires O(1) number of key comparisons to prevent double-spending fraud before it occurs in a digital currency payment transaction.

The search time of O(1) (that is, constant time) means that the hash table data structure used by [23] digital currency payment model requires the same amount of time to prevent double-spending fraud before it occurs in a payment transaction regardless of the input size when compared to the other two considered data structure techniques used by [14]and [19], where the search time to prevent double-spending fraud before it occurs increases as the input size increases.

The comparison of the three data structure techniques used for the implementation of different digital currency payment models in terms of input size n and search time is further illustrated in Table 1, where n is the number of units of digital currency identifications presented for a payment transaction, and the accompanying graph in Figure 9 that shows pictorially the computational time performance of the three data structure techniques.

Table 1: Comparison of the selected data structure techniques

S/n	N	Blockchain T1 = (k*$\log_2$n)	Binary search tree T2 = O($\log_2$n)	Hash table T3 = O(1)
1	0	0.0000	0.0000	1
2	5	23.2193	2.3219	1.0000
3	10	33.2193	3.3219	1.0000
4	15	39.0689	3.9069	1.0000

5	20	43.2193	4.3219	1.0000
6	25	46.4326	4.6439	1.0000
7	30	49.0689	4.9068	1.0000
8	35	51.2928	5.1293	1.0000

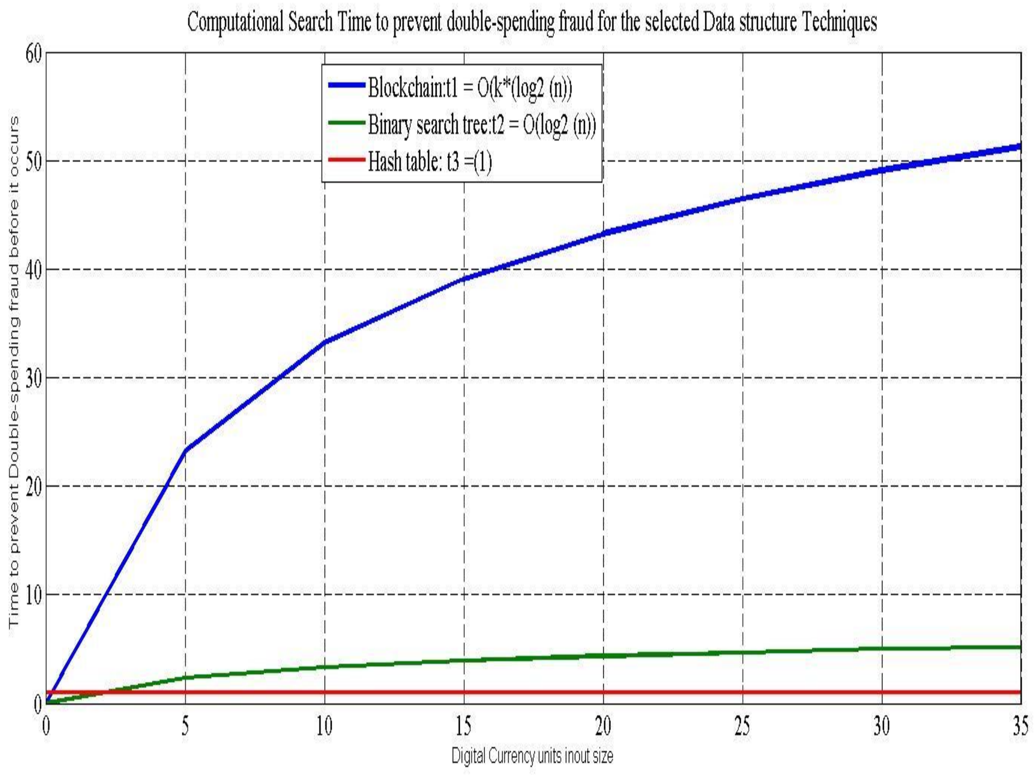


Figure 9: Computational time performance of three data structure techniques

7.0 Conclusion

Every data structure technique has its' own characteristic features that can be exploited for a specific implementation. An appraisal of the three data structure techniques used to prevent double-spending fraud in digital currency payment models in payment transactions show that the hash table data structure has the best search time to prevent double-spending fraud before it occurs. Though hash table data structure has excellent search time to prevent double-spending fraud, most applications developers still prefer blockchain implementation as a result of its' immutability and decentralized nature.

References

[1] Turban, E., King, D., Mckey, J., Marshall. P., Lee, J., & Vielhand , D. (2008). Electronic commerce: A managerial perspective. Pearson Education Ltd, London, 545 – 554.

[2] Raja, J., and Senthil, M. V. (2008). E-payments: Problems and prospects. Journal of Internet Banking and Commerce, 13(11), 200 – 216.

[3] Dejan, S. (2005). Reducing Fraud in Electronic Payment Systems, Proceeding of the 7th Balkan Conference on Operational Research, BACOR 05, Constanta, May, Romania

[3] Sumanjeet, S. (2009). Emergence of payment systems in the age of electronic commerce: The state of the art. Asian Pacific Journal of finance and Banking Research, 3(3), 188 – 197.

[4] Mckay, J., Lee, J., and Vielhand, D. (2008). Electronic Commerce: A managerial Perspective. Pearson Educational Ltd, London, 554 – 556.

[5] Lowry, P. B., Taylor, W., Gregory, D. M., Sean, H., & Degan, K. (2006). Online payment gateways used to facilitate e-commerce transactions and improve risk management. Communications of the Association for information Systems, 17(6), 21 – 38.

[6] Sullivan R., J. (2010). The Changing Nature of Card Payment Fraud: Industry and Public Policy Options, Payment System Research Brief.

[7] Anderson, M. (2010). Fraud: The Facts", Association for Payment Clearing Services (APACS).

[8] Morgan, C. (2013). World payment reports. Capgemini analysis, 23 -29.

[9] Nashidi, T., Miyazaki, S., kouichi, S. (2011). Security analysis of E-Cash systems with malicious insider. Journal of Wireless Mobile Networks, Ubiquitous Computing, and depending applications.

[10] Chokhani, E., Carlisle, A., and Lloyd, S. (2010). Internet X.509 Public Key Infrastructure: Certificate Policy and Certification Practices Framework. Orion Security Solutions, Inc.

[11] Sean, H., Taylor, W., Gregory, D., and Degan, K. (2006). Online Payment Gateways used to facilitate e-commerce transactions and require risk management. Communication of the Association of information Systems, 17(6), 41 – 48.

[12] Chaum, D., (1983). Blind signature for untraceable payments. Proceeding of the Annual International Cryptology conference on Advances in Cryptology (CRYTO'82), Santa Barbara, California.

[13] Nashidi, T., Miyazaki, S., kouichi, S. (2011). Security analysis of E-Cash systems with malicious insider. Journal of Wireless Mobile Networks, Ubiquitous Computing, and depending applications.

[14] Satoshi, N., (2008). The genesis of Crypto Revolution: The Bitcoin White Paper. Journal of Cryptology, 3(2), 99 – 111.

[15] Koblitz, N., and Menezes, A., (2016). Cryptocash, Cryptoccurrencies and Cryptocontracts: Design, Codes and Cryptography. Journal of Cryptology, 78(1), 87 – 102.

[16] Merkle, R., C., (2015). Protocols for Public key Cryptosystems. Journal of IEEE Computer Society, 6(2), 122 – 133.

[17] Osipkor, E., Hopper, N., and Kin, Y. (2007). Combating double-spending using co-operative P2P systems. 27th International conference on distributed computing systems (ICDCS '07), IEEE computer society.

[18] Becker, G., (2008). Merkle Signature Schemes, Merkle Trees and their Cryptanalysis. Journal of IEEE Computer Society, 8(3), 89 – 97.

[19] Yanling, H., Haibin, W., Xuguanf, C., and Xia, L. (2014). Efficient divisible E-cash based on the P-Signature. International Journal of Multimedia and Ubiquitous Engineering, 9(10), 153 – 168.

[20] Heger, D. (2004). A disquisition on the performance behaviour of binary search tree data structures. European Journal for the Informatics Professionals, 5(5), 67 – 75.

[21] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. (2009). Introduction to Algorithms (3rd ed.). Massachusetts Institute of Technology. pp. 253–280. ISBN 978-0-262-03384-8.

[22] Zhang, J., Jia, Y., (2019). Redis rehash optimization based on machine learning. Journal of Physics, Conference Series. 1453: 3.

[23] Aigbe, P. and Onibere, A., (2015). An Immediate Real Time Detection and Prevention of Double-spending fraud in Digital Currency Payment Model. International Journal of Computer Applications, 122(18), pp. 32 – 39.