وزارة التعليم العالي والبحث العلمي
جامعة ميسان
كلية التربية الاساسية

Ministry of Higher Education and Scientific Research
University of Misan
College of Basic Education

Misan Journal for Academic Studies
Humanities, social and applied sciences

مجلة ميسان للدراسات الاكاديمية

العلوم الانسانية والاجتماعية والتطبيقية

ISSN (Print) 1994-697X
(Online)-2706-722X

المجلد23 العدد 51 أيلول 2024
Sep 2024 Issue 51 Vol23

مجلة ميسان للدراسات الاكاديمية

العلوم الإنسانية والاجتماعية والتطبيقية

كلية التربية الأساسية/ جامعة ميسان/العراق

Misan Journal for Academic Studies

Humanities, social and applied sciences

College of Basic Education/University of Misan/Iraq

ISSN (Print) 1994-697X (Online) 2706-722X

المجلد (23) العدد (51) أيلول (2024)

SEP 2024 ISSUE51 VOL 23

www.misan-jas.com

journal.m.academy@uomisan.edu.iq

رقم الايداع في دار الكتب والوثائق بغداد 1326 في 2009

الصفحة	فهرس البحوث	ت
16 - 1	**Using of the Two Tools Analytic Hierarchy Process (AHP) and Technique for Order Performance by Similarity to Ideal Solution (TOPSIS) in Multi-Criteria Decision Making** Ali Dawood Salman Al-khanaqini Omar Mohammed Nasser alashary	1
26 - 17	**Investigation of piranha solution on Color Stability of heat cure acrylic** Noor Azher Al-Rubaie Aseel Mohameed Al-Khafaji	2
33 - 27	**Beamforming and Resource Allocation for Heterogeneous Bands in 6G** Mustafa N. Mnati	3
51 - 34	**The Sounds of Colours: A Pragmatic Study of Toni Morrison's The Bluest Eye** Rafah Abdulkareem Adham Hassan Muayad Hamid	4
60 - 52	**Diagnostic Accuracy of GCF IL33 and sST2 For Periodontitis Stage I, II, and III** Samar A. Abood Ayser N. Mohammed	5
76 - 61	**The Globalization of Human Rights: A Critical Assessment** Omowonuola OKUNNU Bimbo OGUNBANJO	6
87 - 77	**Impact of erythritol air polishing in supportive periodontal care: literature review** Muhsin Kadhim Abbas Hayder Raad Abdulbaqi	7
102 - 88	**Distributed Random Number Generation Fair and Reliable for Blockchain Applications** Mays Moneer Abd Ali Bashar M.Nema	8
114 - 103	**Adhesion of 3d printed acrylic resin with silicone soft liner after sandblast surface treatment (A Review of Literature)** Mohammed Taqi S. Wadi Bayan S. Khalaf	9
125 - 115	**Effect of Immersion Time of a Hydrogen Peroxide and Vinegar Mixture Solution on the roughness of Heat-cured and CAD/CAM Polymethyl Methacrylate Resin** Ahmed I.Shanef Firas Abdulameer Farhan	10
135 - 126	**Evaluation the Salivary levels of Interleukin-23 in Individuals with Thyroid Disorders** Riam Hassoun Harbi Maha Adel Mahmood	11
150 - 136	**The Interplay Between Chronic Kidney Disease and Periodontal Health: A Comprehensive** Ahmed M Mahdi Maha Sh Mahmood	12
157 - 151	**Ethics In Orthodontic Clinical Practice: (A review article)** Zainab Mousa Kadhom Alaa Faleh Albo Hassan Shaymaa Shaker Taha	13
172 - 158	**Preparing of Controlled Release Systems for Atenolol and Studying it is in Vitro Dissolution and Swelling** Mohammed R . Abdul - Azeez	14
179 - 173	**Evaluation the salivary *anti-Porphyromonas gingivalis* (IgA and IgG) response in relation to sera levels of Ferritin and Vitamin D in Patients with Beta-Thalassemia Major** Shahad fayiz abd Maha Adel Mahmood	15
188 - 180	**The role of different adjunctive plaque control modalities in orthodontic patients with gingivitis** Ola Issam majed Raghad Fadhil abbas	16

205 - 189	**Assessed performance of E-learning methods and benefit from (AI) for Undergraduate Engineering stage** Ali Dhahir Alramadhan	17
224 - 206	**The use of counterpoint in Kurdish orchestral compositions: (Dilshad Muhammad Saeed)** ary kadir Mohammed Seerwan Mohammad Mustafa	18
248 - 225	**Demographic Transitions and Spatial Reallocation of Population Variables in the Administrative Territories of Al-Majar Al-Kabeer District Over the Period 1997-2022** Ahmed Sihoud Hashem	19
263 - 249	**The economic and social reasons leading to the phenomenon of child labor in Al-Fuhud district in the year 2023** Osama Hameed Majeed	20
277 - 264	**Exposure to Iraqi satellite channels and its role in shaping public attitudes toward political organizations** Khuzaymah Nizar Khudair	21
294 - 278	**The Situational Implicture of Al-Sararqus in the Structural and Stylistic Correlation** Karima Abed Jumaa	22
314 - 295	**Isolate and Identification of Rhizoctonia solani Kühn causing Eggplant rot and accompanying bacteria with toxicity studying of fungicides and their effect in vitro** Alaa Hassan Al-Farttoosy Fatima Chasib Bader	23
331 - 315	**A Geopolitical Approach to the City ; with Emphasis on the Central Elements of Power and Competition** Hasan Kamran Dastjerdi Narjessadat Hosseini Abbas Ghali Al-Hadithi	24
347 - 332	**Patterns of Multiple Intelligences among High School Students in Misan Governorate According to The (TEEN-MIDAS) Scale and Their Relationship to Achievement** Haneen Mahdi Siker Ahmed Abdul-Muhsin	25
364 - 348	**The Impact of Transitional Regulations on the Application of National Reconciliation and National Amnesty Mechanism (A Comparative Study)** Hadi Hassan Kashash AlRikabiu Sayyid Ali Mirmad Najafabad	26
381 - 365	**Technical Diversity in Contemporary Kurdish Art** Sirwan Rafat Ahmed	27
389 - 382	**The interpretive approach at Al-Shahrastani** Mohammed Reda Salman Mohammed Ali Tajery Reda Moadeb	28
409 - 390	**The Degree of Consistency Between Mathematics Teachers' Beliefs Towards Constructivist Learning and Their Practice of its Skills** Haider Abdulzahra Alwan	29
420 - 410	**Hadiths posted about Imam Ali (peace be upon him) on social media platforms and their impact on society’s behavior** Lectuer.Mohammed Jabbar Jassim Ali	30
442 - 421	**The stylistic diversity that transformed the form and concept of contemporary visual arts.** Rajaa Kareem Jiboori Alobaidi	31

ISSN (Print) 1994-697X
ISSN (Online) 2706-722X

DOI:
https://doi.org/10.54633/2333-023-051-008

Distributed Random Number Generation Fair and Reliable for Blockchain Applications

Mays Moneer Abd Ali[1] Bashar M.Nema[2]
[1.2]Informatics Institute for Postgraduate Study.
Department of Computer Science ,mustansiriyah University. Iraq
[1]nmnm44mnmn@gmail.com
[2]Bmn774@uomustansiriyah.edu.iq
https://orcid.org/0009-0000-7532-4788

Abstract:

Random number generators, or RNGs, are essential in a variety of domains, such as blockchain applications, information technology, and computer security. Pseudorandom numbers, their classification, experimental analysis, and the search for cryptographically secure pseudo-random number generators (CSPRNGs) are the main topics of this paper's investigation into cryptographic random number generators. The paper highlights the technical difficulty of producing tamper-proof, unexpected, and verifiable random numbers in distributed systems like blockchains, drawing on findings from a Chainalysis study.

Applications of blockchain emphasize the need for safe random numbers, especially in the gaming and token (NFT) industries, which are valued at over $44 million. Platforms such as the blockchain lottery Pooltogether, which has received over $10 million from players and pays out over $8,000 in weekly prizes at random, show the lottery's worldwide appeal. The Proof-of-Stake (POS) consensus method used by Ethereum, which makes use of decentralized RNGs, demonstrates how safe.

Keywords: *Random Number Generators, Cryptographic Security, Blockchain Applications, Pseudorandom Numbers, Decentralized Systems.*

Intoducation:

one of study deals Random sequence generators are utilized in various fields like computer security, information technology, mathematical simulation, sampling, and random variant production, despite the difficulty of obtaining random sequences due to computers' deterministic nature (Almaraz et al.,2023) . The paper investigates random number generators for cryptographic applications, focusing on pseudorandom numbers, their generators, classification, experimental setup analysis, cryptographically secure pseudo-random number generators, and research findings, based on a Chainalysis study (Werapun et al.,2023). One of the most important technological problems in distributed systems, including blockchains, is the generation of tamper-proof, unexpected, and verifiable random numbers Blockchain gaming and token (NFT) sectors are valued at over

$44 million, with NFTs given randomly to creators at different rarities. Pooltogether, a blockchain-based lottery system, receives over $10 million from gamers worldwide, with weekly awards totaling over $8,000 distributed at random. The Ethereum Proof-of-Stake (POS) consensus mechanism selects a miner to add data to the chain. The blockchain is powered by a decentralized Random Number Generator (RNG) (Almaraz et al.,2023),(Werapun et al.,2023).

Random numbers are crucial for daily life and are increasingly important in the open Internet environment. They are tamper-proof and public, allowing everyone to confirm the protocol. Traditional methods require large samples and low efficiency, while new technologies can reduce time. However, current random functions are insecure, making it urgent for computers to develop more secure, reliable, and unpredictable random numbers without achieving complete randomness(Almaraz et al.,2023, Werapun et al.,2023).

Random numbers are crucial for daily life and online transactions. Generating them securely is essential for information and network security. Traditional methods require large samples and low efficiency. The Internet can reduce random number generation time, but current pseudo-random functions are insecure(Werapun et al.,2023).

Research on network random number security focuses on pseudo-random functions to protect user data while minimizing time-consuming methods like rolling dice or drawing lots (Fatemi et al.,2023), Nakamoto introduced Bitcoin's foundational principle, allowing individuals to conduct transactions without central authority mediation or approval. The use of a specific formula for generating random numbers is unsafe, highlighting the need for more reliable, random, and unpredictable options for computers (Fatemi et al.,2023),(Hsieh et al.,2022).

Holotescu later termed this concept "disintermediation" to signify the absence of a central authority within a blockchain system (Hsieh et al.,2022). Disintermediation occurs through blockchain, a peer-to-peer network where nodes store, transmit, and validate transactions. Scientists and technologists are interested in developing random number generators, which should produce uniformly distributed and independent numbers. True random number generators (TRNGs) pass specific tests for specific applications(Barakbayeva et al.,2024, Mannalatha et al.,2023), Inherently random sources, like atmospheric or thermal noise, are recommended for critical applications due to their skewed nature and the challenge of certifying their quality(Mannalatha et al.,2023).

Inherently random sources, like atmospheric or thermal noise, are recommended for critical applications due to their skewed nature and the challenge of certifying their quality (Bezuidenhou et al.,2023, Simić et al.,2020).

Random number generators (RNG) are used in cryptographic applications for encryption and decryption. Deterministic random bit generators (DRBGs) and nondeterministic random bit generators (NRBGs) are used. NRBGs rely on physical processes like thermal noise and radioactive decay, but are expensive, time-consuming, and hardware-dependent. DRBGs, based on deterministic processes, produce pseudo-random bits with ideal statistical properties, making them integral to cryptographic systems and statistical simulation testing (Mannalatha et al.,2023, S., Kumar et al.,2020).

Saku and colleagues utilized the N_choice framework technique to generate random numbers for smart contracts, achieving 129 times faster results than random bit generators(Sako et al.,2022).

Related Works:

The literature review will explore decentralized random number generators in blockchain technology and parallel computing environments. It will discuss the importance of uniform random numbers in parallel computing, their management, and the specific requirements for decentralization, verifiability, and scalability in blockchain applications. The review will highlight the need for secure, verifiable, and independent streams of random numbers. A lot of different RNG algorithms available for blockchain are present in the literature. Here, we give a brief overview of just a few of these approaches. As far as we know, this secret random number generation problem is not addressed on previous workaround (Huang et al.,2024).

Hashes or timestamps are commonly used in smart contracts to generate random numbers, preventing manipulation by the caller. However, this method is vulnerable to exploitation by miners and lacks uniform distribution. Despite these vulnerabilities, it remains prevalent in Ethereum smart contracts(He, L et al.,2024).

Oracles act as intermediaries in smart contracts, transferring real-world information onto the blockchain. However, this centralized approach is susceptible to manipulation and fails to meet security standards, necessitating adjustments to the formula used(Huang et al.,2024),(He, L et al.,2024).

RANDAO is a method that uses commitment schemes and open participation to generate random numbers. Players commit by selecting a nonce and recording the hash in a smart contract. The random number is determined by the xor of all inputs. However, this method is vulnerable to manipulation(Chen et al.,2024)

Verifiable delay functions are used in protocols to prevent tampering by requiring sequential computation and preventing parties from predicting or manipulating the value of inputs Verifiable delay functions are used in protocols to prevent tampering by requiring sequential computation and preventing parties from predicting or manipulating the value of inputs These functions, given inputs b1, . . . , bn, take considerable time to compute r := f(b1, . . . , bn), preventing parties from predicting r and strategically choosing their inputs to manipulate its value(Lee, S et al.,2024).

Verifiable random functions (VRFs) are used in proof-of-stake protocols like Algorand and Ouroboros for local computation and proof verification. However, they may not be suitable for our scenario, where the secret value must belong to CASSIE. Schindler introduces Hydrand, a distributed random beacon protocol built on Publicly Verifiable Secret Sharing, which ensures continuous output without reliance on a trusted party, even in adversarial environments(Du, Z et al.,2024).

Das critiques existing studies for lack of security guarantees, specificity, and high costs. They propose SPURT, a secure method for network nodes with one-third malicious nodes. SPURT synchronizes beacon generation across nodes, enabling independent computation of beacon values. It can produce 84 beacons per minute in a 32-node network(Chen et al.,2024),(Du, Z et al.,2024).

Bhat introduces RandPiper, a protocol for generating random beacons in scenarios with node turnover. It focuses on five key requirements: unpredictability, unbiased output, optimal resilience, low communication overhead, and efficient node replacement. Bhat proposes a resilient Byzantine fault-tolerant (BFT) state machine replication protocol, GRandPiper and BRandPiper, to address predictability issues and improve beacon generation speed (Afzaal et al.,2024).

Problem Statment:

The goal its challenges of achieving scalability in blockchain applications, focusing on parameters like block size, transaction processing speed(Dhulavvagol et al.,2024), and block generation rate., revealing limitations in scope, depth, and methodology for compare protocol families or consensus approaches, they often overlook other dimensions of scalability(Amores et al.,2024). Additionally focusing only on throughput and storage. Existing where, methodologies lack systematic approaches and comprehensive coverage(Amores et al.,2024),(Nguyen et al.,2023). In contrast aims for previous literatures , considering both horizontal and vertical scalability dimensions systematically many problems , with the goal of achieving depth and breadth to solve any problem in real-world block chain applications (Koukaras et al.,2023).

Methodology:

This paper reviews recent studies on decentralized random number generation, focusing on proposed techniques and their questions about each approach to address the problem of random number generation.

1. Stage one create and enveonment:

The proposed method involves actors, algorithmic techniques, and main characteristics, using Python libraries and functions to perform specific tasks. Here is an explanation of each library:

For example, an integrated work environment from multiple offices was used to create this work and express the results. For example, hashlip was used to calculate free funds. We also used time to interact and measure time periods during implementation. Also, Random is considered one of the important libraries for generating random numbers and controlling their pattern. As well as I used math for approximation and all mathematical functions from square roots and others, I used tkniter to build graphical interfaces, while ttk represents an important library for compatibility and integration with different operating systems to make graphical interfaces consistent, and finally I used canvas to draw shapes and texts on interface elements. As we show below flowchart in Figure (1), generating random numbers system.

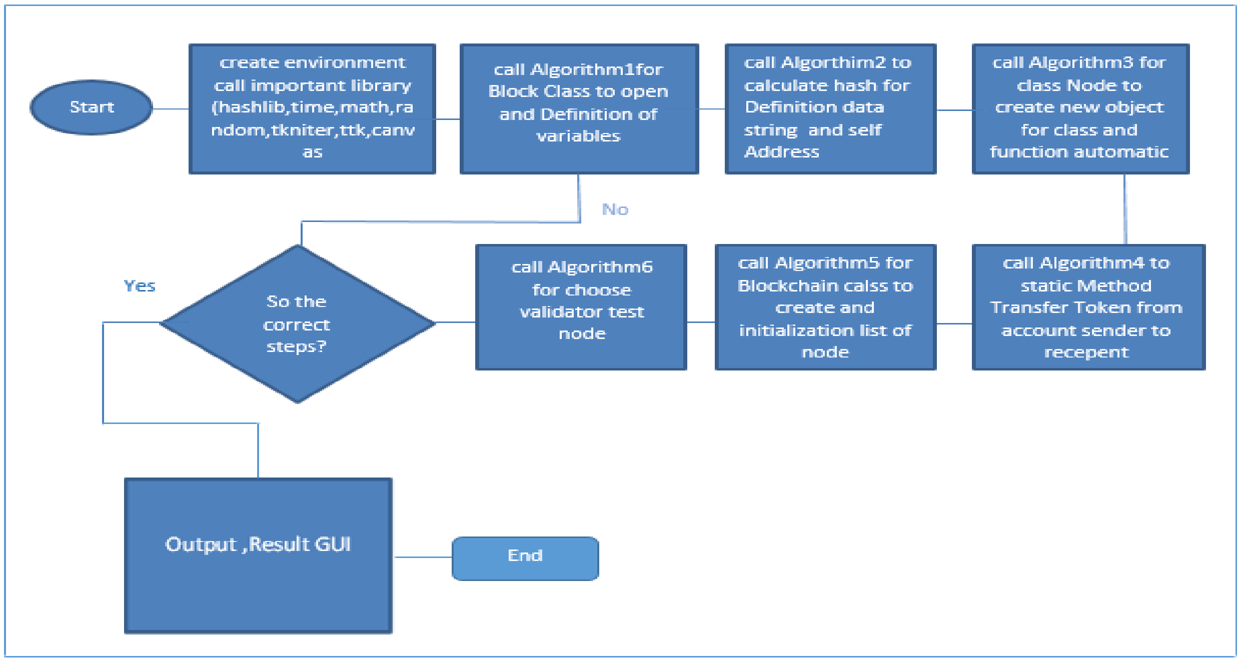

Start
create environment call important library (hashlib,time,math,random,tkniter,ttk,canvas
call Algorithm1for Block Class to open and Definition of variables
call Algorthim2 to calculate hash for Definition data string and self Address
call Algorithm3 for class Node to create new object for class and function automatic
call Algorithm4 to static Method Transfer Token from account sender to recepent
call Algorithm5 for Blockchain calss to create and initialization list of node
call Algorithm6 for choose validator test node
So the correct steps?
No
Yes
Output ,Result GUI
End

Figure (1): a general environment for generating a random numbers system.

2. Stage two prposed system and built Algorithms:

The code defines a Block class in Python, representing a block in a blockchain, essential for distributed technology applications like cryptocurrencies. It contains data like index, timestamp, previous_hash, validator, balance, block_type, and hash.

```
Inetalization  Block class Step 1: Open class Block Step 2: Define variables: - self - index - timestamp - data - previous_hash - validator - balance_before - balance_after - block_type Step 3: Initialize variables in constructor: - self.index = index - self.timestamp = timestamp - self.data = data - self.previous_hash = previous_hash - self.validator = validator - self.balance_before = balance_before - self.balance_after = balance_after - self.block_type = block_type Step 4: End class .
```

Figure (2) Description of the Block Class Algorithm in Pseudocode

In Figure (3), prposed system the calculate_hash function uses the SHA-256 algorithm to calculate block hashes by combining block information into a single string, resulting in a unique digital signature for each block.

```
Algorithm Calculate Hash
   Step 1: Open class Block
   Step 2: Define function calculate_hash(self)
      - data_string = combine block attributes (index, timestamp, data, previous_hash,
validator, balance_before, balance_after, block_type) into a single string
   Step 3: Call the hashlib library to use SHA-256
      - hash_result = hashlib.sha256(data_string.encode()).hexdigest()
   Step 4: Return hash_result
End Algorithm
```

Figure (3), Pseudocode for Calculate Hash Algorithm SHA-256

This pseudocode Figure(4): defines a Node class used in blockchain networks to represent individual nodes. The __init__ function is a constructor that initializes each node with an address and balance, automatically called when a new Node object is created. This setup allows each node to store and access its unique address and balance.

```
Algorithm Class Node
    Step 1: Open class Node
    Step 2: Define variables:
        - self
        - address
        - balance
    Step 3: Define constructor __init__(self, address, balance)
        - self.address = address
        - self.balance = balance
    Step 4: Create a new object for the class
    Step 5: Automatically call the constructor function
End Algorithm
```

Figure(4): Pseudocode for Node Class Algorithm"

which is commonly used in blockchain applications to represent nodes in the network. The _init_(self, address, balance) function is a constructor for a class, automatically calling when a new object is created. This code defines a class called Node It assigns the address and balance values to the new object, allowing access to the node's address and balance. This code defines a class representing a node in the blockchain network. Node: It is the name of the class, and it is used to represent a node in the network.

Figuer (5) show pseudocode defines a static method transfer_tokens that transfers tokens from one account (sender) to another (recipient) if the sender has a sufficient balance. If the sender's balance is not enough, it returns False. Additionally, it defines a static method generate_chaotic_random_number that generates a random number between 1 and 100, which can be used for various purposes.

```
Algorithm staticmethod
    Step 1: Define static method transfer_tokens(sender, recipient, amount)
        - If sender.balance >= amount:
            - Proceed with the transfer
            - Return True
        - Else:
            - Return False indicating insufficient funds
    Step 2: Define static method generate_chaotic_random_number()
        - Generate a chaotic random number within the range of 1 to 100
        - Return the random number
    Step 3: End Algorithm
```

Figure(5): Pseudocode for staticmethod Operations"

Where Static methods in Python enable the creation of functions within a class that are independent of instance variables. For instance, transfer_tokens(sender, recipient, amount) can facilitate token transfers between senders and recipients, effectively managing token transactions within the class structure.

The Blockchain class manages a chain of blocks using properties like chain, nodes, and smart contract, which handle and execute financial transactions or commands within the blockchain structure.as we see in figure (6) wher Blockchain Initialization and Genesis Block Creation Algorithm.

```
Step 1: InitializeBlockchain()
    - Initialize a new blockchain instance with a genesis block, an empty list of nodes, and a smart contract instance.
  Step 2: CreateGenesisBlock()
    - Define a function to create the genesis block.
    Step 3: ReturnGenesisBlock()
    - Return a new instance of the Block class initialized as the genesis block.
    Step 4: End Algorithm
```

Figure(6):Blockchain Initialization and Genesis Block Creation Algorithm

The Blockchain class manages the chain of blocks through properties like chain, nodes, and smart_contract, which execute financial transactions or commands.

```
Step 1: choose_validator()
    - Validate and check nodes.
        - If self.nodes list is empty:
            - Return None since there are no nodes available.
Step 2: Compute total_balance by summing up the balance attribute of all nodes in self.nodes.
Step 3: Call self.smart_contract.generate_chaotic_random_numbers(10) to generate a list of 10 chaotic random numbers.
Step 4: If no suitable validator is found after iterating through all nodes, return None.
Step 5: End Algorithm
```

Figure(7):choose_validator Method Algorithm

This algorithm in figure (7) defines the choose_validator method, which is responsible for selecting a validator node from the self.nodes list within the Blockchain class. Here's a breakdown of its steps:

Where in first step Validation and Node Check , Checks if the self.nodes list is empty. If empty, returns None indicating no available nodes.after that, Compute Total Balance, Computes the total_balance by summing up the balance attribute of all nodes in self.nodes and Generate Chaotic Random Numbers for Calls self.smart_contract.generate_chaotic_random_numbers(10) to generate a list of 10 chaotic random numbers, last step Validator Selection ,where If no suitable validator is found after iterating through all nodes, it returns None.

This method facilitates the selection of a validator node based on predefined criteria and uses chaotic random numbers to introduce randomness into the selection process, crucial for maintaining security and fairness in blockchain operations.

3. Blockchain Application with Smart Contracts and UI Integration

The function choose_validator selects a validator for network operations or blocks by checking node availability, calculating total balance, generating random numbers, shuffles, comparing nodes, and returning None if no validator is found.

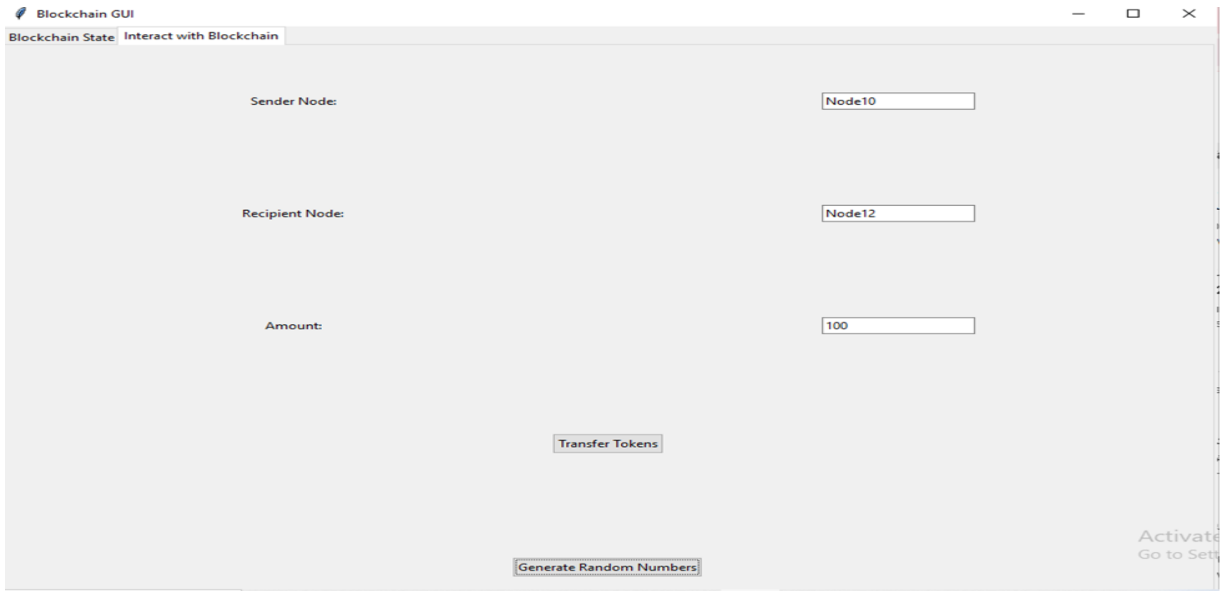
The application comprises Block, Node, SmartContract, CircularChainCanvas, Blockchain, and BlockchainApp classes, representing blocks, nodes, contracts, and the chain network, with BlockchainApp serving as the main interface.

In general, the application allows the user to interact with the blockchain, they can perform balance transfers between nodes and execute arbitrary contracts, and the application also displays the details of the blocks visually in the user interface.

Here we created a blockchain using Python and linked smart contracts to the blockchain, where we created two smart contracts. The first is responsible for exchanging currencies (Transfer Tokens) between the nodes within the blocks, and the second is responsible for generating random numbers (Generate Random Numbers).Using the Chaotic Random Number algorithm . After that, we developed an application that controls the smart contract so that the smart contract is called at any time through an interface. We also made from the same application another interface that performs operations and events, visualizing the blockchain so that all events appear in the form of blocks and arrows that appear as graphics when... Code execution.

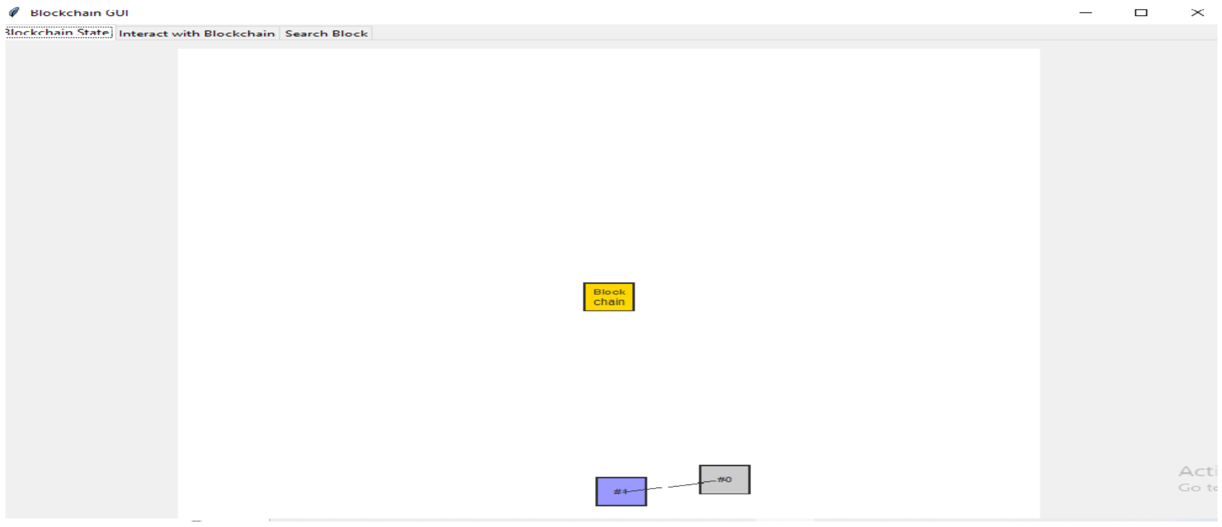
4. Stage Three Discussions and Results:

Using Python and its libraries, a number generator, Transfer Token, and Blockchain were created. The Blockchain can be used as a display of the central block and the interface block that was created during the generation process of the Blockchain, but the search feature allows users to search on text to enter the block number or display information in it, which is represented by (indexing, Printer, data, previous hash), as we can see in Figure 8, where there are three commands referring to each of them. The random generation process on the blockchain interface performs the effects that initially contain the central or main blockchain and an empty string (Block0, genesis). However, after the creation of the blockchain, the first connected block is interrupted, appearing as if it were continuing with the empty chain, as completed in Figure 9.



Figuer(8) : Blockchain Display with Searchable Block Information(GUI)

The program executes by pressing the Run button, displaying a code execution interface and a graphical blockchain user interface, generates 10 random numbers, and calculates sender and recipient balances.



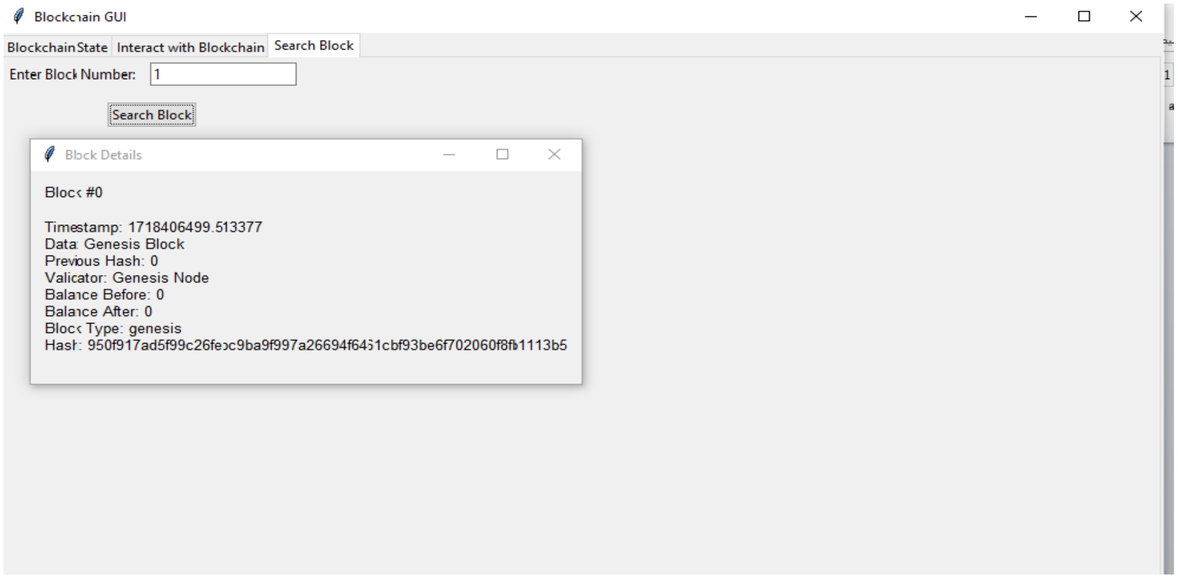
Figuer(9) Blockchain Display State

The Blockchain graphical user interface contains three windows, each of which performs a specific function and they are as follows:

1.Blockchain State: It is the window for executing the program or displaying the execution process as graphics.

2.Search Block: The proposed system allows for the process of inquiring about the Blockchain through this window and through the Search Block Command. When you enter the required block number and search for it, all the data associated with this block will appear sequentially. This can be seen in Figure No. 10.

3.Interact with Blockchain: It is considered one of the most important stages of the proposed system, as it contains two basic stages. The first stage is the process of transferring the account or amount between the sender and the recipient, based on smart contracts and all its procedures related to deletion, creation, and addition in terms of increase and decrease in the transferred amount. The second stage is the process of generating random numbers for smart contracts in a way Random, confidential and secure,This can be seen in Figure No.9_1.



Block #0

Timestamp: 1718406499.513377
Data: Genesis Block
Previous Hash: 0
Validator: Genesis Node
Balance Before: 0
Balance After: 0
Block Type: genesis
Hash: 950f917ad5f99c26fe3c9ba9f997a26694f6451cbf93be6f702060f8fb1113b5

Figuer(9_1) Search Block

The process of generating random numbers and creating a new blockchain enables block inquiries by entering its number and pressing the Search Block button.

The program executes by pressing the Run button, displaying a code execution interface and a graphical blockchain user interface, generates 10 random numbers, and calculates sender and recipient balances.

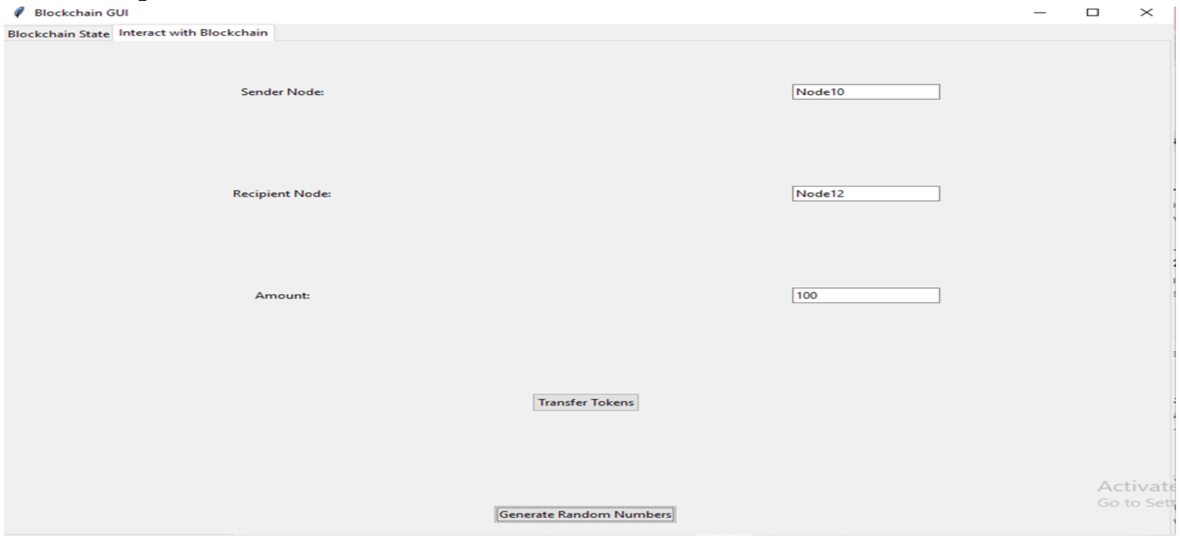


Figure11:Interact With Blockcha

(Table-1): Table of results of the first implementation of generating random numbers and Transfer Tokens.

NO	No.Node	S.Ch.RN	S.B.B	S.B.A	R.B.B	R.B.A
1	Node16	484	66	66	87	87
2	Node1	538	66	66	89	89
3	Node3	397	67	67	89	89
4	Node20	497	67	67	89	89
5	Node12	622	67	67	89	89
6	Node15	531	67	67	89	89
7	Node20	558	67	67	89	89
8	Node16	588	67	67	89	89
9	Node6	531	67	67	89	89
10	Node15	327	67	67	89	89

As shown in Table No. 1 and according to Figure No. 4, the amount was transferred from node 10 to node 12, and the amount is 100. When the program is executed, random numbers will be generated after the Transfer Tokens process. The total of random numbers that have a certain Threshold that cannot be exceeded will be calculated, which here equals 577, according to the table. First, the value of the amount is calculated by the sender and recipient before and after the process of transferring the amount to random numbers.

(Table-2): Table of results of the first implementation of generating random numbers and Transfer Tokens. represents the second itration

NO	No.Node	S.Ch.RN	S.B.B	S.B.A	R.B.B	R.B.A
1	Node3	519	66	66	86	86
2	Node16	577	66	66	86	86
3	Node6	477	66	66	86	86
4	Node10	325	66	66	86	86
5	Node16	469	66	66	86	86
6	Node5	564	66	66	86	86
7	Node8	478	66	66	86	86
8	Node4	559	66	66	86	86
9	Node2	337	66	66	86	86
10	Node20	463	66	66	87	87

The table (Table2) depicts the results from the second iteration of a process involving random number generation and token transfers within a blockchain system. Each row in the table corresponds to a specific transaction between nodes, detailing various aspects of the transaction:

Where in the columns:Receiver Balance Before (R.B.A) and Receiver Balance After (R.B.B): These columns show the balance of the recipient node before and after receiving tokens, respectively.

Sender Balance Before (S.B.A) and Sender Balance After (S.B.B): These columns indicate the balance of the sender node before and after sending tokens, respectively.

Sender Chosen Random Number (S.Ch.RN): This column displays the chaotic random number selected by the sender node for executing the transaction.

Node Number (No.Node): Identifies the specific nodes involved in each transaction.

Transaction Number (NO): Represents the sequential order of transactions.

These results illustrate how tokens are transferred between nodes in the blockchain network, with balances updated accordingly before and after each transaction. The use of chaotic random numbers adds a layer of randomness and security to the transaction process, ensuring fairness and unpredictability in node selection and token transfers within the blockchain ecosystem.

(Table-3): Table of results of the first implementation of generating random numbers and Transfer Tokens. represents the others itration

NO	No.Node	S.Ch.RN	S.B.B	S.B.A	R.B.B	R.B.A
1	Node6	405	64	64	79	79
2	Node20	516	64	64	79	79
3	Node12	504	64	64	79	79
4	Node4	535	64	64	79	79
5	Node18	527	64	64	79	79
6	Node17	451	64	64	79	79
7	Node5	616	64	64	79	79
8	Node15	576	64	64	79	79
9	Node10	602	64	64	79	79
10	Node15	348	64	64	79	79

In the results presented in all Tables(1,2,3), we observe several key aspects:

Balance Changes: The table shows how the balances of both the sender and receiver nodes change before and after each transaction (R.B.A, R.B.B, S.B.A, S.B.B). This reflects the transfer of tokens between nodes in the blockchain network.

Random Number Selection: The "Sender Chosen Random Number (S.Ch.RN)" column indicates the chaotic random numbers chosen by the sender node for each transaction. This randomness enhances the security and fairness of the transaction process within the blockchain system.

Node Identification: Each transaction is associated with specific nodes identified by "Node Number (No.Node)". This helps track which nodes are participating in the transactions.

Transaction Sequence: The "Transaction Number (NO)" column provides the sequential order of transactions, showing the chronological execution of token transfers within the blockchain network.

Together, these components show how the blockchain system tracks and records token transfers, identifying the nodes participating in each transaction for accountability and transparency, and managing them using randomization for security.

The research article included the following symbols to indicate different outcomes and processes during the implementation:

No.Node: In the blockchain system, a node's number or identification designates its participation in a transaction or operation.

total for ch.RN: The total amount of chaotic random numbers generated or used in the course of the operations.

S.B.B. stands for "Sender Balance Before" and describes the sender node's balance before a transaction is carried out.
S.B.A. stands for "Sender Balance After" and indicates the sender node's balance following the completion of a transaction.
The symbol for "Recipient Balance Before" is R.B.B.

These symbols are used throughout the paper to describe and analyze the outcomes and behaviors of the blockchain system during the execution of operations such as token transfers and random number generation. They provide a standardized way to refer to specific data points and variables essential for understanding the implementation and results presented in the research.

Conclusion:

In this study, we utilized a set of standardized symbols to effectively convey the outcomes and processes observed during the implementation of blockchain operations. These symbols, including No.Node for node identification, Sum of ch .RN for cumulative chaotic random numbers, and S.B.B, S.B.A, R.B.B, R.B.A for sender and recipient balances before and after transactions, and CH.RN for sets of chaotic random numbers, provided a structured framework for analyzing and presenting data.Through the application of these symbols, we were able to systematically document the behavior of the blockchain system in managing token transfers and random number generation. This approach facilitated a clear understanding of transactional dynamics, node interactions, and the role of randomness in enhancing security within the blockchain network.

Moving forward, the consistent use of these symbols not only contributed to the clarity and coherence of our findings but also established a foundation for future research and comparative analyses in blockchain technology. By standardizing our reporting with these symbols, we aim to promote transparency, reproducibility, and advancement in the field of blockchain implementation and analysis.

Referances

Almaraz Luengo, *E., & Román Villaizán, J. (2023). Cryptographically Secured Pseudo-Random Number Generators: Analysis and Testing with NIST Statistical Test Suite. Mathematics, 11(23), 4812*. **https://doi.org/10.3390/math11234812**

Werapun, W., Karode, T., Suaboot, J., Arpornthip, T., & Sangiamkul, E. (2023). NativeVRF: A Simplified Decentralized Random Number Generator on EVM Blockchains. *Systems*, *11*(7), 326. **https://doi.org/10.3390/systems11070326**

Fatemi, P., & Goharshady, A. (2023, October). Secure and decentralized generation of secret random numbers on the blockchain. In *2023 Fifth International Conference on Blockchain Computing and Applications (BCCA)* (pp. 511-517). IEEE. 10.1109/BCCA58897.2023.10338880

Hsieh, C. H., Yao, X., Zhang, Q., Lv, M., Wang, R., & Ni, B. (2022). BCsRNG: A Secure Random Number Generator Based on Blockchain. *IEEE Access*, *10*, 98117-98126. 10.1109/ACCESS.2022.3206450

Barakbayeva, T., Cai, Z., & Goharshady, A. (2024, May). SRNG: An efficient decentralized approach for secret random number generation. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. https://hal.science/hal-04518059/

Mannalatha, V., Mishra, S., & Pathak, A. (2023). A comprehensive review of quantum random number generators: Concepts, classification and the origin of randomness. *Quantum Information Processing*, *22*(12), 439. https://doi.org/10.1007/s11128-023-04175-y

Bezuidenhout, R., Nel, W., & Maritz, J. M. (2023). Permissionless blockchain systems as pseudo-random number generators for decentralized consensus. *IEEE Access*, *11*, 14587-14611. 10.1109/ACCESS.2023.3244403

Simić, S. D., Šajina, R., Tanković, N., & Etinger, D. (2020, September). A review on generating random numbers in decentralised environments. In *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)* (pp. 1668-1673). IEEE. 10.23919/MIPRO48935.2020.9245302

Kalanadhabhatta, S., Kumar, D., Anumandla, K. K., Reddy, S. A., & Acharyya, A. (2020). PUF-based secure chaotic random number generator design methodology. *IEEE transactions on very large scale integration (VLSI) systems*, *28*(7), 1740-1744. 10.1109/TVLSI.2020.2979269

Sako, K., Matsuo, S., & Mori, T. (2022, July). Distributed Random Number Generation Method on Smart Contracts. In *Proceedings of the 2022 4th Blockchain and Internet of Things Conference* (pp. 1-10). https://doi.org/10.1145/3559795.3559796

Huang, C., Song, R., Gao, S., Guo, Y., & Xiao, B. (2024). Data Availability and Decentralization: New Techniques for zk-Rollups in Layer 2 Blockchain Networks. *arXiv preprint arXiv:2403.10828*. https://doi.org/10.48550/arXiv.2403.10828

He, L. (2024). A Comparative Examination of Network and Contract-Based Blockchain Storage Solutions for Decentralized Applications. *arXiv preprint arXiv:2401.07417*. https://doi.org/10.48550/arXiv.2401.07417

Chen, Y., Hu, D., Xu, C., & Chen, N. (2024). An annotation assisted smart contracts generation method. *IEEE Access*. 10.1109/ACCESS.2024.3386751

Lee, S., Jee, E., & Lee, J. (2024). Implementation Study of Cost-Effective Verification for Pietrzak's Verifiable Delay Function in Ethereum Smart Contracts. *arXiv preprint arXiv:2405.06498*. https://doi.org/10.48550/arXiv.2405.06498

Du, Z., Li, Y., Fu, Y., & Zheng, X. (2024). Blockchain-based access control architecture for multi-domain environments. *Pervasive and Mobile Computing*, *98*, 101878. https://doi.org/10.1016/j.pmcj.2024.101878

Afzaal, H., Zafar, N. A., Tehseen, A., Kousar, S., & Imran, M. (2024). Formal Verification of Justification and Finalization in Beacon Chain. *IEEE Access*. 10.1109/ACCESS.2024.3389551

Dhulavvagol, P. M., Totad, S. G., Anagal, A. M., Anegundi, S., Devadkar, P., & Kone, V. S. (2024). ShardedScale: Empowering Blockchain Transaction Scalability with Scalable Block Consensus. *Procedia Computer Science*, *233*, 432-443. https://doi.org/10.1016/j.procs.2024.03.233

Amores Sesar, I. (2024). *Scaling the Unscalable: A Study About Consensus* (Doctoral dissertation, Universität Bern).*https://crypto.unibe.ch/archive/theses/2024.phd.ignacio.amores.sesar.pdf*

Nguyen, H. T. (2023). A transition towards decentralized service market: blockchain-based enablers, challenges, and solutions. https://urn.fi/URN:ISBN:9789526238906

Koukaras, P., Afentoulis, K. D., Gkaidatzis, P. A., Mystakidis, A., Ioannidis, D., Vagropoulos, S. I., & Tjortjis, C. (2024). Integrating Blockchain in Smart Grids for Enhanced Demand Response: Challenges, Strategies, and Future Directions. *Energies*, *17*(5), 1007. **https://doi.org/10.3390/en17051007**

Al-Hammash-Haitham, M. S. H. (2024). Enhancement methods of intrusion detection systems using artificial intelligence methods (TLBO) Algorithm. *(Humanities, social and applied sciences) Misan Journal of Academic Studies*, *23*(49), 105-112.

Hamady, I. K. A. (2023). A Secure Search for Outsourced Image Collection Based Content-Based Image Retrieval. *(Humanities, social and applied sciences) Misan Journal of Academic Studies*, *22*(47), 348-359.

Majeed, H. L. (2023). Using neural networks to solve image problems through artificial intelligence. (Humanities, social and applied sciences) Misan Journal of Academic Studies, 22(47), 409-416

خلاصة:

تعد مولدات الأرقام العشوائية، أوRNGs ، ضرورية في مجموعة متنوعة من المجالات، مثل تطبيقات blockchain، وتكنولوجيا المعلومات، وأمن الكمبيوتر. تعد الأرقام العشوائية الزائفة وتصنيفها وتحليلها التجريبي والبحث عن مولدات الأرقام العشوائية الزائفة الآمنة تشفيرًا (CSPRNGs) هي الموضوعات الرئيسية التي تتناولها هذه الورقة البحثية في مولدات الأرقام العشوائية المشفرة. تسلط الورقة الضوء على الصعوبة التقنية في إنتاج أرقام عشوائية مقاومة للتلاعب وغير متوقعة ويمكن التحقق منها في الأنظمة الموزعة مثل البلوكشين، بالاعتماد على نتائج دراسة تشيناليسيس.

تؤكد تطبيقات blockchain على الحاجة إلى أرقام عشوائية آمنة، خاصة في صناعات الألعاب والرموز(NFT) ، والتي تقدر قيمتها بأكثر من ٤٤ مليون دولار. تظهر منصات مثل يانصيبblockchain Pooltogether ، الذي تلقى أكثر من ١٠ ملايين دولار من اللاعبين ويدفع أكثر من ٨٠٠٠ دولار كجوائز أسبوعية بشكل عشوائي، جاذبية اليانصيب في جميع أنحاء العالم. توضح طريقة إجماع إثبات الملكية (POS) التي تستخدمهاEthereum ، والتي تستخدم RNGs اللامركزية، مدى أمانها.