Performance Evaluation of the AuRa Consensus Algorithm for Digital Certificate Processes on the Ethereum Blockchain

Robiah Arifin[1]*, Wan Aezwani Wan Abu Bakar[2]*, Mustafa Man[3], Evizal Abdul Kadir[4]
Department of Big Data, Infostructure and Network Management Centre, Universiti Sultan Zainal Abidin, 21030 Kuala Nerus, Terengganu, Malaysia[1]
Faculty of Informatics and Computing-Universiti Sultan Zainal Abidin, 22200 Besut Campus, Besut, Terengganu, Malaysia[2]
Faculty of Computer Science and Mathematics-Universiti Malaysia Terengganu, 21030 Kuala Nerus, Terengganu, Malaysia[3]
Universitas Islam Riau, Jl. Kaharuddin Nasution 113, Pekanbaru 28284, Riau, Indonesia[4]

***Abstract*—The blockchain serves as a distributed database where data is stored across different servers and networks. It encompasses various types, with Bitcoin, Ethereum, and Hyperledger being notable examples. To safeguard the security of data transactions on the blockchain, it relies on a consensus algorithm. This algorithm facilitates agreement among nodes within the network. There are multiple types of consensus algorithms, each possessing unique specialties and characteristics. This paper drives into the examination of specific Authority Round, here claimed as AuRa_ori consensus algorithm. The AuRa_ori is a specific type of PoA consensus mechanism used primarily in private or permission blockchain networks. It works by having a set of trusted validators take turns in a round-robin fashion to produce new blocks. It is supported by Parity and Ethereum Clients. AuRa_ori assumes that all the authority nodes are synchronised and honest on every transaction process. In AuRa_ori, every transaction process will execute the four phases i.e., assigning of a new leader, proposing a block, commencing agreement and finally, the phase of committing. However, there exist some discrepancies in some of the phases. In response to the scenario, this paper presents a thorough discussion on the vulnerabilities adhered in AuRa phases in transaction execution by focusing on the first phase of assigning a new leader and the third phase, namely the agreement. The vulnerabilities are subjected to the risk of impacting the performance of Transaction Speed Per Second (TPS), Transaction Throughput (TGS), Percentage Decrease (PD) of TPS and Percentage Increase (PI) of TGS. The new improved method, named AuRa_v1 is parallel presented to overcome the vulnerabilities of AuRa_ori at the selected phases. It aims to increase the TPS and to calculate the PD in transaction process using the Ethereum private blockchain systems. The implementation used three set of data scroll certificate. The result showed that the AuRa_v1 able to decrease the TPS almost 30% based on difference number set of data.**

***Keywords*—Blockchain; Ethereum; consensus algorithm; smart contract; AuRa_ori; AuRa_v1**

I. Introduction

The section drives the important components within the scope of the paper discussion on the AuRa_ori algorithm starting from the introduction of the blockchain technology, the platform that offers the technology, to what component and algorithm that determines the efficiencies of the blockchain technology. The efficiency determinants of the blockchain relies on the robustness of the consensus algorithm, wherein the discussion on the AuRa_ori is initiated. The consensus algorithms are crucial in further bolstering security within blockchain transactions by orchestrating an agreement among nodes. This ensures that the throughput remains consistent, uniform, and valid across the network.

Blockchain, as a decentralised distributed database, serves as a platform for storing transactional data and information. It represents a departure from traditional methods, transitioning data management from a single centralised location to a decentralised setup across various servers or networks [1]. This decentralised data storage comprises a series of interconnected records known as blocks [2].

Data is dispersed among different nodes across diverse servers within the blockchain network. Often referred to as a peer-to-peer (P2P) network, blockchain operates through nodes communicating directly with each other [3]. Utilising cryptographic techniques and hash functions, blockchain ensures the security and integrity of digital data stored within its system [4].

While blockchain encompasses several types, Bitcoin, Ethereum, and Hyperledger stand out as the most popular and widely used variants [5], [6]. These platforms have been developed over the years with a focus on encouraging businesses to integrate blockchain into their operations [7]. Despite their similarities, each blockchain platform possesses unique characteristics. Ethereum, conceptualized by Vitalik Buterin in 2013 [8], emerged as a response to the limitations of Bitcoin [9]. Crowdfunding in 2014 facilitated the advancement of Ethereum's technologies, leading to its live network launch in 2015 [10].

Ethereum implements distributed data storage, enabling users to deploy their applications and operate their blockchain instances [11]. Notably, Ethereum offers the capability to store data with an unlimited block size [12]. Specifically, the AuRa_ori consensus algorithm is used in voting domain and focusing on the electronic voting application. The researcher was comparing the AuRa_ori and Geth based on certain criteria such as time, consistency, and scalability. This application runs the test net environment including remix IDE, Ethereum test net and web3.

Additionally, the efficiency of blockchain is reinforced by the consensus algorithm, which facilitates an agreement process among nodes dispersed across various servers and networks. This algorithm ensures the maintenance of consistency and decentralisation among nodes within the blockchain system [4],[13],[14]. The consensus mechanism necessitates four essential elements in transactions, as outlined in Table I.

TABLE I. Element of consensus algorithm

Element	Description
Termination	The successful transaction process dependence on the content of block that proposed
Agreement	The acceptance block needed to assign by the honest authorities
Validity	The block should be accepted if nodes that received is valid and same with the proposed block
Integrity	Only the honest node should be acceptance their transaction

There are numerous types of consensus algorithms that are able to support the security of data transaction in blockchain environment. However, this paper provides the detailed exposure of the consensus algorithms in conjunction to the strengths and the weaknesses in AuRa_ori used in Ethereum and reveals and proposes the improvement mechanism as a means of solution on the weaknesses.

There are numerous types of consensus algorithms that are able to support the security of data transaction in blockchain environment. However, this paper aim discussed the detail AuRa_ori including it procedure to transact data into the blockchain. We also provide the detailed exposure of the consensus algorithms in conjunction to the strengths and the weaknesses in AuRa_ori used in Ethereum environment.

Prior to the weakness of AuRa, the analysis of the effect of AuRa weakness in term of transaction speed was executed. The weakness is able to expose the risk including the cloning attack and malicious leader. Then the improvement mechanism as a means of solving weaknesses were proposed and revealed.

II. AURA_ORI CONSENSUS ALGORITHM

AuRa_ori, short for Authority Round, was initially proposed for the Parity client, which utilises Rust as its programming language [15]. This consensus relies on the assumption of honest authorities and synchronous network communication among nodes. If any issues such failed to assign a new leader or leader failed to produce the signature the process must be repeat at beginning. This impact the throughput and transaction process. Additionally, AuRa_ori consists of four key steps including assigned a new leader, proposed block, agreement and commit transaction into the blockchain. This consensus mechanism has been implemented by various platforms including Laava, VecHain Thor, xDai DPOS network, Microsoft Azure (for deployment only), and Kovan Testnet [16].

AuRa_ori has gained widespread adoption in blockchain applications, with nearly 4,000 projects implementing this consensus algorithm [17]. For example, AuRa_ori was implemented in health domain focused on health record sharing. The researcher proposed new methods to improve the time in block transaction process of AuRa_ori [18]. The AuRa_ori is well implemented to support the Copyright System in [19]. This effort drives the implementation of AuRa_ori and performs the analysis of its transaction throughput. It also concerns about the network synchronisation.

A. Assigning a New Leader

Initially, a leader is assigned, with each authority taking turns using the Round Robin Algorithm [20]. The leader assignment process involves calculating the difference value $t_d)$ between the current timestamp t and the last timestamp at the genesis t_g as shown in Eq. (1).

$$t_d = t - t_g \tag{1}$$

Where t_d is a difference values of t and t_g . While t is a current timestamp and t_g is the last time stamp at the genesis. After getting the values of t_d, next is to assign the leader among the authorities based on Eq. (2).

$$i = \frac{t_d}{D} mod n \tag{2}$$

In the formula, i represents the node tasked with assigning a leader, while D stands for the Step Duration specified in the genesis file. The n denotes the number of nodes configured in the blockchain environment. Following the assignment of the leader based on this formula, the subsequent step involves the leader proposing a block.

B. Propose Block

The block that proposed by a leader based on set of values, there are data v on t_p , number of proposed blocks n and signature of authorities account on leader v_s. Every proposed block process must include the set value $\{v, n, v_s\}$. Based on set values the other authorities will apply the voting process.

C. Agreement

After finding the leader and proposing the block $\{v, n, v_s\}$, the next step is agreement among the nodes. The block that was proposed agreed or not by the authorities depends on data on proposed block v, numbering of proposed block n and the authorities account on leader is different v_s. After getting the agreement result from the authorities, The data will transfer to block transaction queue that declare as t_q. Then after the authorities 'members complete their agreement process, the next step is the voting process based on Eq. (3).

$$2f + 1 \leq n \tag{3}$$

Where f is a faulty node, n is a number of nodes. In that case the number of faulty nodes at least must be less or the same as the number of nodes 50%.

D. Commit

Subsequently, the leader puts forward a block, comprising a leader signature key, current timestamp, and block data. Following the proposal of the block, the process proceeds to the voting and acceptance phase. If the agreement achieves more than 50%, the next is the commit process. Thet_q will be inserted into the blockchain, and the value of t_q and $t_p$will be deleted from the transaction process. Fig. 1 shows the patterns of AuRa in committing the transaction data.

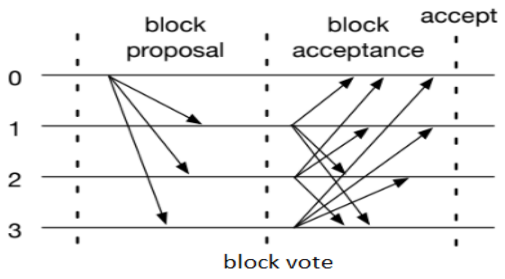


Fig. 1. Pattern of aura transaction process.

According to Fig. 1, the AuRa_ori process is bifurcated into two primary stages: block proposal and block acceptance. Once the leader is designated, the leader initiates the block proposal phase by proposing a block. Subsequently, the block acceptance phase commences, during which the members of the authority vote on the block proposed by the leader to determine if it should be committed.

III. AURA_ORI RELATED WORKS AND DRAWBACKS

The implementation of AuRa_ori can be widely seen in various domains of applications. It is considered as a stable algorithm and able to support the transaction process into the blockchain. However, the AuRa_ori is vulnerable to the exposure of attacks if the issues persist in all over the transaction process.

A. Challenges Faces in AuRa

The AuRa_ori has been widely embraced to enhance the speed of blockchain transactions. Previous research has delved into various aspects of AuRa_ori. For instance, [21] examined partitioning tolerance in AuRa_ori. The author investigates strategies for preventing partitioning tolerance in AuRa_ori. The AuRa_ori is capable of detecting only the absence status of authorities.

However, it cannot discern whether missing authorities are inactive or located in a different partition, as it lacks the ability to identify network partitioning. Lack of ability to identify availability authorities 'impact to transaction speed and throughput. It needed extra time if authorities are detected not available a long transaction process because AuRa_ori needed to assign a new leader and get the signature key from a new leader to assign data into the blockchain. Extra time was risk to any attack.

Additionally, the study in [22] conducted research on transaction speeds using the Geth and Parity clients. Both Geth and Parity support by PoA, with Geth focusing on Clique and Parity on AuRa_ori. On average, the Parity client exhibited a 91% faster transaction speed compared to Geth. The experiment based on transaction data from 1000 to 5000. The comparison is conducted on a private test net, considering factors such as CPU, RAM, and the number of nodes. The performance analysis criteria included the time, consistency and scalability. However, it does not consider the security and safety criteria in this analysis.

Furthermore, the study in [23] discussed a comparison between Geth and Parity in terms of transaction speed and performance level. Their analysis cantered on the consumption of CPU and RAM resources by the two Ethereum clients. However, this research did not specify the type of consensus utilised, and details regarding CPU, RAM, and server type were not provided. As previously noted, AuRa_ori operates under the assumption that nodes remain synchronised, which impacts both availability and consistency. According to prior research by [24], which focused on comparing AuRa_ori, Clique, and PBFT in terms of availability, consistency, and partition tolerance, the synchronisation requirement of AuRa poses a risk, especially when deployed over a wide area. This algorithm relies heavily on accurate timestamps for both genesis and proposed blocks.

Regarding transaction speed, [25] explored the performance of AuRa_ori in comparison to PoW. The research discussed how AuRa surpasses PoW in transaction speed due to its lower computing power requirements and higher throughput. Moreover, AuRa_ori offers the enhancement in security compared to PoW, as it incorporates an additional layer of security and operates with two levels of acceptance: the proposed block and the subsequent voting process but the voting process exploit the security issue if it needed extra time to complete the task.

B. Reviews on the Challenges that AuRa_ori Faced

The widespread implementation of AuRa_ori has sparked interest among researchers to scrutinise its intricacies. Previous studies have identified several drawbacks associated with AuRa_ori, particularly in the initial step of leader assignment and the subsequent voting agreement phase.

Assigning a new leader for each transaction process poses a challenge within AuRa_ori. To accomplish this, AuRa_ori employs a Round Robin schedule. However, if authority members are unavailable, the process of reassigning a new leader becomes necessary, resulting in decreased throughput, affecting transaction speed and time execution [19]. Because AuRa needs to assign a new leader and it increases time to complete the transaction. Then it needs to create a new fork to allow a queued transaction in proposing a new block.

Process to assign a new leader causing risk to attack including AuRa_ori assumes each node is synchronised, but sometimes unexpected network delays happen. To overcome this situation AuRa_ori allowed the authorities to postpone the validation process and cause the block to be delayed. This exploited loophole to assigned the malicious leader when the previous transaction not completed because time delay to complete the transaction.

Furthermore, complications arise if the assigned leader fails to create a signature or becomes corrupted, leading to decreased throughput and failed transactions [26]. To mitigate these issues, the concept of a dummy signature has been proposed. If authority members accept the dummy signature, it is converted into a real signature then the transaction process is able to continue. The challenge is how to verify the dummy signature is accurate. Because this research did not explain how the dummy signature creates and verify.

Additionally, changes in assigned authorities can occur at any time, any changes requiring agreement from other authorities. Normally the changes occur if the nodes not available because of network or configuration issues. Assigning new authorities process introduces delays until the new authorities are accepted, impacting the security, performance, and transaction speed of AuRa_ori.

The election of a leader for each transaction process exposes the algorithm to potential attacks [27] because the leader needed to create the signature. The signature was used in proposed block, voting committing transactions. If the leader failed to sign a signature the new leader must assign and it impacts the execution time, throughput, transaction speed and epoch time. Additionally, the validation process among authorities before committing transactions to the blockchain requires extra time to achieve agreement, affecting throughput and potentially leading to forks.

A delayed verification can also leave AuRa_ori vulnerable to cloning attacks (CA) [28]. This occurs due to the creation of forks during delayed verification, allowing for the replication of identities using the same private and public keys. Delay verification occur if the leader failed to sign the signature or the signature not valid. Then it needed to assign new leader and new signature, then impact the execution time and potential the attacks.

To address these challenges, this paper proposes the implementation of a heartbeat-based approach to thwart cloning attacks. This involves signing and sending the heartbeats to other authorities, who then accept proposed blocks based on the heartbeat signature.

IV. Proposed Methodology

Based on previous studies, AuRa_ori is tightly bound with the transaction speed issues. These issues because of the procedure in AuRa_ori needed to assign a new leader for every transaction process and needed the verify phase before the transaction was able to commit into the blockchain. This research proposed a new algorithm called AuRa_v1 to overcome these issues.

The new proposed algorithm named AuRa_v1 consists of 4 steps including preassigned new leader, transaction pending, proposed block and voting. Numbering steps are the same with the AuRa_ori, but it's needless to assign a new leader for every transaction process. Each step in AuRa_v1 is enhanced from the AuRa_ori.

A. Predefine Leader

Predefining a leader is a process to assign a new leader as oppose to AuRa_ori where it requires the assignment of a new leader at every transaction process. In contradiction, AuRa_v1 assigns a new leader only on condition the leader that was assigned is not available, or the signature key by the previous leader is not valid. Fig. 2 highlights the detail of predefined leader procedure.

Algorithm 1

```
[1] While true do
[2]     $t_d$ ← TIMESTAMPDIFF(second, $t - t_g$)
[3]     $l$ ← (($t_d/D$) mod $n$)
[4]     $sk$ ← generate by $l$
[5]     Transaction_Pending($sk, l, n$)
[6] End Process
```

Fig. 2. AuRa_v1 predefined leader algorithm.

The new leader assignment in Fig. 2 will be made available if and only if either the occurrence of the 2 conditions i.e., 1) the previous assigned leader is not available, or 2) the signature key owned by the previous assigned leader is not valid. Line 2 aims to get the difference value between current timestamp and genesis timestamp. The difference value based on Eq. (4).

$$t_d = t - t_g \tag{4}$$

Where value of t_d is a difference value of current timestamp t and genesis time, t_g. However, this research proposes the difference value was converted into a second based on timestamp format using the function $TIMESTAMPDIFF(second)$. We propose converting into second because it is able to improve the transaction speed. Then the value of t_d was used to choose the new leader as line 3 based on Eq. (5),

$$l = \frac{t_d}{D} mod n \tag{5}$$

Where l is new leader that was assigned and D is the value of step duration, D value can be any number in minutes or second, normally value of D setup by developer including 5 until 10 seconds. Then n is a number of authorities that register in blockchain setup.

A. Voting and Commit Transactions

Voting and commit process is the last process before the transaction is stored into the blockchain. This process there is no need to collect the agreement from the available authorities. It just counts the number of available authorities. Fig. 3 illustrates the voting and commit procedure.

Algorithm 2

```
[1]  voting($b_p, sk, l, n$)
[2]  $nt$ ← datetime.now()
[3]  If(($nt - t$)<= $D$)then
[4]    $ca$ ← count($aa$)
[5]    if $ca$ >= 1 ∧ $ca$ <= $n$) then
[6]       Commit
[7]    else
[8]       Failed
[9]  else
[10]    Failed
[11] If ($l \triangleq l$)then
[12]    transaction_pending($sk, l$)
[13] else
[14]    pre_assigned_leader()
[15] End
```

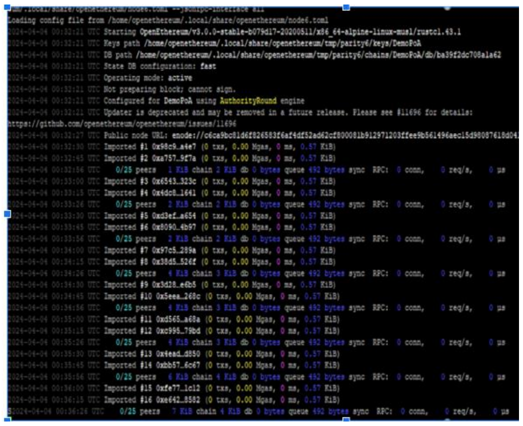
Fig. 3. Algorithm of voting and commit procedure.

In Fig. 3 at line 2 to 3, ntrefer to new current timestamp. This procedure needed to capture the new current timestamp. Then the new timestamp minus t and the value must be less or the same with step duration value D. Line 4 to 6, it counts the authorities available, if authorities available more than 1, and numbers of available authorities less or the same with the number of authorities the transaction process is able to commit into the blockchain. Lastly line 7 until 14, refer to failure completed the prerequisite, the procedure back to transaction pending or pre assigned leader process.

V. Implementation

This studies aims to analyst the performance of AuRa_ori and AuRa_v1. The analysis based on result of digital certificate process. The certificate process implemented into two version including AuRa_ori and AuRa_v1. To support the implementation of AuRa_ori and AuRa_v1, the installation and configuration of AuRa_v1 and AuRa_ori was executed. The installation and configuration of AuRa_ori based on standard AuRa_ori. Fig. 4 portrays the installation and configuration of the AuRa_ori.

However the installation and configuration of AuRa_v1 based on proposed algorithm of AuRa_v1. Then the node file was created for both of AuRa_ori and AuRa_v1. The number of nodes file based on the number of node that is created on AuRa_ori and AuRa_v1 engine. Fig. 5 depicts part of nodes file.



```
um/.local/share/openethereum/node6.toml --jsonrpc-interface all
Loading config file from /home/openethereum/.local/share/openethereum/node6.toml
2024-04-04 00:32:21 UTC Starting OpenEthereum/v3.0.0-stable-b079d17-20200511/x86_64-alpine-linux-musl/rustc1.43.1
2024-04-04 00:32:21 UTC Keys path /home/openethereum/.local/share/openethereum/tmp/parity6/keys/DemoPoA
2024-04-04 00:32:21 UTC DB path /home/openethereum/.local/share/openethereum/tmp/parity6/chains/DemoPoA/db/ba39f2dc708a1a62
2024-04-04 00:32:21 UTC State DB configuration: fast
2024-04-04 00:32:21 UTC Operating mode: active
2024-04-04 00:32:21 UTC Not preparing block; cannot sign.
2024-04-04 00:32:21 UTC Configured for DemoPoA using AuthorityRound engine
2024-04-04 00:32:21 UTC Updater is deprecated and may be removed in a future release. Please see #11696 for details:
https://github.com/openethereum/openethereum/issues/11696
2024-04-04 00:32:27 UTC Public node URL: enode://c6ca9bc81d6f826583f6af4df52ad62cf800081b912971203ffee9b561496aec15d98087618d04
2024-04-04 00:32:30 UTC Imported #1 0x98c9…a4e7 (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:32:45 UTC Imported #2 0xa757…9f7a (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:32:56 UTC    0/25 peers   1 KiB chain 2 KiB db 0 bytes queue 492 bytes sync  RPC:  0 conn,    0 req/s,    0 µs
2024-04-04 00:33:00 UTC Imported #3 0x6543…323c (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:33:15 UTC Imported #4 0x4dc8…1641 (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:33:26 UTC    0/25 peers   2 KiB chain 2 KiB db 0 bytes queue 492 bytes sync  RPC:  0 conn,    0 req/s,    0 µs
2024-04-04 00:33:30 UTC Imported #5 0xd3ef…a654 (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:33:45 UTC Imported #6 0x8090…4b97 (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:33:56 UTC    0/25 peers   2 KiB chain 2 KiB db 0 bytes queue 492 bytes sync  RPC:  0 conn,    0 req/s,    0 µs
2024-04-04 00:34:00 UTC Imported #7 0x97c5…289a (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:34:15 UTC Imported #8 0x38d5…526f (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:34:26 UTC    0/25 peers   4 KiB chain 3 KiB db 0 bytes queue 492 bytes sync  RPC:  0 conn,    0 req/s,    0 µs
2024-04-04 00:34:30 UTC Imported #9 0x3d28…e6b5 (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:34:45 UTC Imported #10 0x5eea…268c (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:34:56 UTC    0/25 peers   4 KiB chain 3 KiB db 0 bytes queue 492 bytes sync  RPC:  0 conn,    0 req/s,    0 µs
2024-04-04 00:35:00 UTC Imported #11 0xd565…a68a (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:35:15 UTC Imported #12 0xc995…79bd (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:35:26 UTC    0/25 peers   4 KiB chain 3 KiB db 0 bytes queue 492 bytes sync  RPC:  0 conn,    0 req/s,    0 µs
2024-04-04 00:35:30 UTC Imported #13 0x4ead…d850 (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:35:45 UTC Imported #14 0xbb57…6c67 (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:35:56 UTC    0/25 peers   6 KiB chain 4 KiB db 0 bytes queue 492 bytes sync  RPC:  0 conn,    0 req/s,    0 µs
2024-04-04 00:36:00 UTC Imported #15 0xfe77…1c12 (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
2024-04-04 00:36:15 UTC Imported #16 0xe642…8582 (0 txs, 0.00 Mgas, 0 ms, 0.57 KiB)
S2024-04-04 00:36:26 UTC    0/25 peers   7 KiB chain 4 KiB db 0 bytes queue 492 bytes sync  RPC:  0 conn,    0 req/s,    0 µs
```

Fig. 4. The installation and configuration of aura engine.

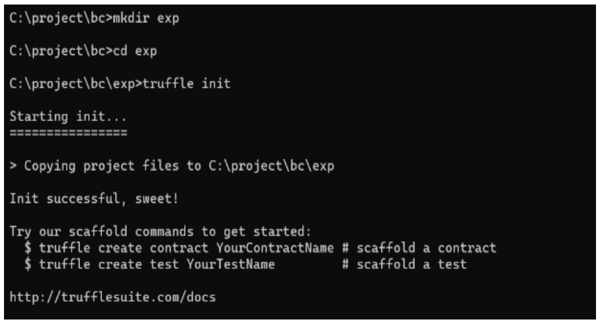
Code1: xxx.toml

```
[1] [parity]
[2] chain = "chain directory"
[3] base_path = "aura engine directory"
[4] [network]
[5] port = 30301
[6] [rpc]
[7] port = 8541
[8] apis=["web3","eth","net","personal","parity","parity_set",
[9] "traces", "rpc", "parity_accounts"]
[10][websockets]
[11]port = 8451
[12][ipc]
[13]disable = true
```

Fig. 5. Code of parity engine configuration

Based on Fig. 5, the file $xxx.toml$ used the parity engine (line 1) and read the chain file (line2). Line 4 and 5 mention the port network that is used is 3301. Line 6 and 7 mention the Remote Procedure Call(RPC) port that used 8541. Line 8 and 9 are about dependencies that implement and line 10 - 11 is about the web sockets port used is 8451.

Besides that, this study also require created the smart contract file. Smart contract is a special element in Ethereum, it consists of the business logic for executing the command of user requirement. After the completed the smart contract, the next step is to migrate the smart contract with the both AuRa_v1 and AuRa_ori engine. The migration process used the truffle framework. The truffle engine is needed for installation and configuration as denoted in Fig. 6 while the three sets of scroll datasets are characterised as in Table II.



```
C:\project\bc>mkdir exp

C:\project\bc>cd exp

C:\project\bc\exp>truffle init

Starting init...
================

> Copying project files to C:\project\bc\exp

Init successful, sweet!

Try our scaffold commands to get started:
  $ truffle create contract YourContractName # scaffold a contract
  $ truffle create test YourTestName         # scaffold a test

http://trufflesuite.com/docs
```

Fig. 6. The truffle framework installation.

TABLE II. SET OF DATA

Set of Data	Number of Data	Size per Data	Type of Data
Set 1	1021	32.8KB - 33.01KB	Scroll
Set 2	2435	32.2KB - 33.11KB	Scroll
Set 3	3422	32.5KB - 33.07KB	Scroll

Referring to Table II, Set 1 includes 1021 data, set 2, 2435 data and set 3 with 3422 data. These data consist information of scroll student including name, program and final cumulative grade. These 3 sets of data are being executed in transaction process on both AuRa_ori and AuRa_v1.

Additionally, the research has also developed the frontend interface to generate the digital certificate. The interface connected with both AuRa_ori, and AuRa_v1 through the Application Programmers Interface (API) and backend code as depicted at Fig. 7.

The front-end consists of a few modules including the dashboard, create certificate and verification. These function able to access regarding the user access level. Fig. 8, Fig. 9 and Fig. 10 show the development interfaces done during experimentation. Regarding Fig. 8, the interface that is developed consists of a few modules including the create certificate and verified certificate. Then Fig. 9, is an interface to create the digital certificate. Lastly at Fig. 10, is an interface to process the digital certificate.

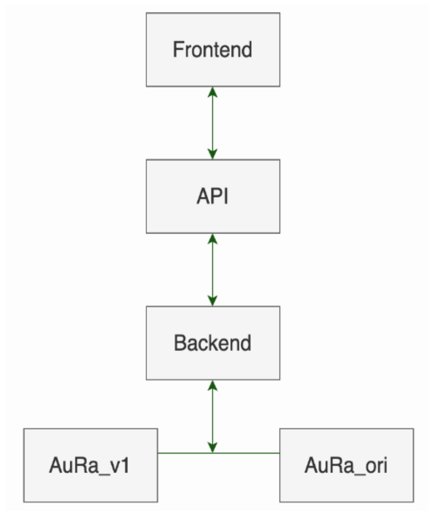


Fig. 7. The framework of digital certificate process.

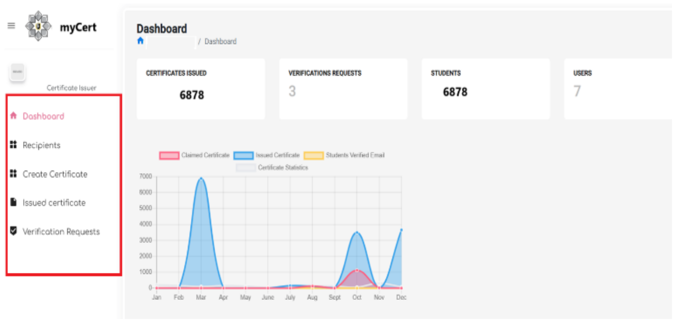
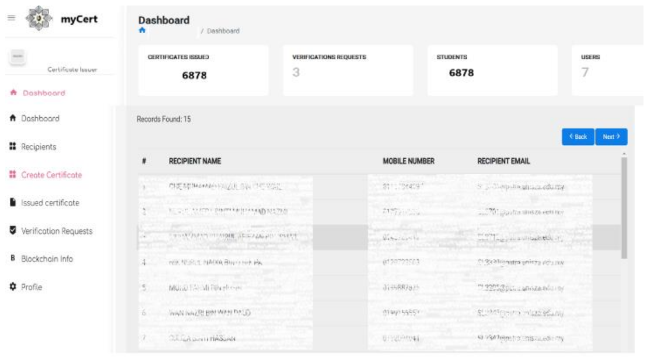


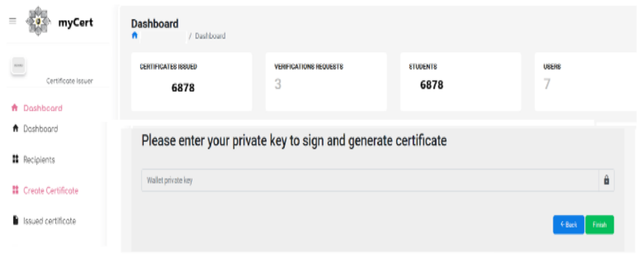
Fig. 8. The dashboard and main interface.



Records Found: 15

#	RECIPIENT NAME	MOBILE NUMBER	RECIPIENT EMAIL
1	[illegible]	[illegible]	[illegible]
2	[illegible]	[illegible]	[illegible]
3	[illegible]	[illegible]	[illegible]
4	[illegible]	[illegible]	[illegible]
5	[illegible]	[illegible]	[illegible]
6	[illegible]	[illegible]	[illegible]
7	[illegible]	[illegible]	[illegible]

Fig. 9. The digital certificate generate interface.



Please enter your private key to sign and generate certificate

Fig. 10. The digital certificate generate process.

VI. Result and Discussion

According these implementation, the TPS result were captured to analysis the performance of AuRa_v1 and AuRa_ori.

A. *TPS Result of AuRa_ori*

This result based on implementation the AuRa_ori used three set of data. Regarding Fig. 11, the TPS result of set data 1 is 0.059. Set data 2 the TPS result is 0.032 and set data 3 is 0.069. According the TPS result, set data 2 more rapidly compared set data1 and set data 3. Set data 3 very slow compared set data 1 and set data 3.

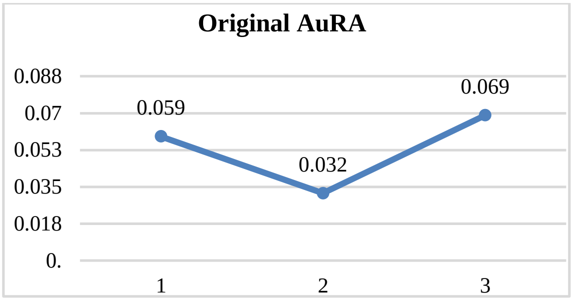


Fig. 11. TPS result of AuRa_ori.

B. *TPS Result of AuRa_v1*

This outcome based on execution the AuRa_v1 used three set of data. Fig. 12 depicted the result of TPS that executed the AuRa_v1. The result of Set data 1 is 0.041, set data 2 is 0.029 and set data 3 is 0.036. These TPS result showed that the set data 2 is faster and set data 3 is slow compare three set of data.

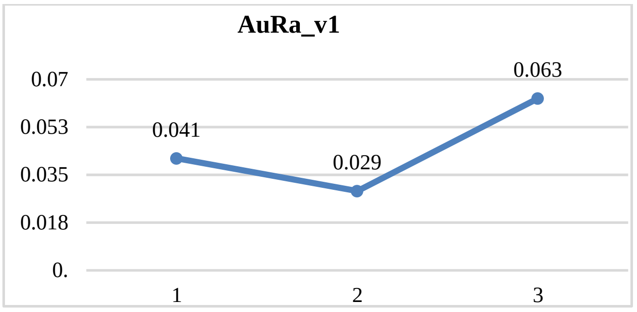


Fig. 12. TPS result based on aura_v1.

C. *TPS AuRa_ori Versus AuRa_v1*

This section aims to compare the result between the AuRa_ori and AuRa_v1. Regarding Fig. 13 depicted that the result of AuRa_ori (orange line) at set data 1 is 0.059, set data 2 is 0.032 and set data 3 is 0.069. Then TPS result of AuRa_v1 (blue line) , set data 1 is 0.041, set data 2 is 0.029 and set data 3 is 0.063.

According these result, set data 1, AuRa_v1 is able decrease the transaction process compared AuRa_ori. Then set data 2 and set data3 also showed that AuRa_v1 prove decrease the TPS result compared AuRa_ori.

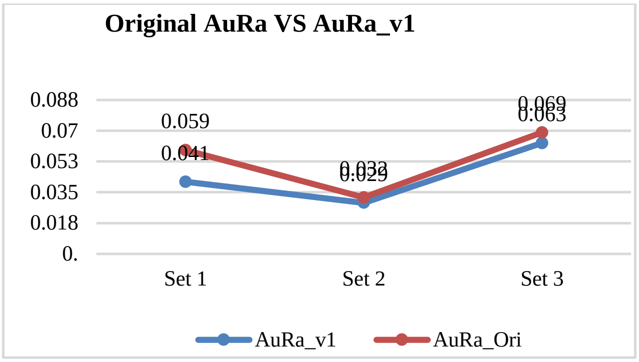


Fig. 13. Comparison TPS result between AuRa_ori and aura_v1.

D. Percentage Decrease of TPS

This research aims to decrease the transaction speed process into the blockchain using the AuRa_v1. Regarding the TPS result, the percentage Decrease PD measured to obtain the PD based on AuRa_v1. The measurement of $PD$the based on Eq. (6),

$$PD = \left(\frac{(i_v - f_v)}{i_v}\right) * 100 \qquad (6)$$

In Eq. (6), PD refer to Percentage Decrease, then i_v is interval value for AuRa_ori value and f_v is the final value for AuRa_v1 value. The PD result was measured and described in Table III.

TABLE III. THE PERCENTAGE DECREASE OF TPS

Data	Result
Set Data: 1	30%
Set Data: 2	9.375%
Set Data: 3	8.69%

The results show that the implementation of AuRa_v1 able to decrease the TPS result. Set data 1, able to decrease the TPS result by 30%, set data 2 by 9.375 % while set data 3 with 8.69%.

VII. CONCLUSION

The optimal consensus algorithm hinges on the shortest transaction execution time within the blockchain. Previous research discussed earlier indicates that both assigning a new leader and voting during the transaction process impact the transaction speed and throughput of AuRa_ori and AuRa_v1. The heightened transaction speed presents risks such as decreased throughput, cloning attacks, and the possibility of assigning a malicious leader.

Nearly all previous studies underscore the limitations of the AuRa_ori consensus algorithm regarding performance of TPS, attributed to the necessity of assigning a new leader for every transaction process. This results in a reduction in transaction throughput. Additionally, the increased TPS of AuRa_ori is influenced by the requirement for a voting phase before transactions can be committed to the blockchain. The future plan is restructured prior to results obtained that aims to measure the TGS and TPS use the same set of data. The compare the result between AuRa_ori and AuRa_v1. Also calculate the PD and PI.

ACKNOWLEDGMENT

We express sincere gratitude to the Center of Research and Innovation Management (CREIM) at UniSZA for their invaluable financial support towards the publication. Our heartfelt appreciation also goes to all team members from UniSZA, including Pn. Robiah Arifin, the PhD candidate for the technical configuration of the project, and Dr. Wan Aezwani Wan Abu Bakar as her supervisor for the conceptual and technical proofread writing, as well as the UMT member, Assoc Prof. Ts. Dr. Mustafa Man for the network linkages and the reviewer's suggestions. Also, we are grateful for the international contributions and collaboration provided by Dr. Evizal Abdul Kadir, a research fellow from Universitas Islam Riau, Indonesia.

REFERENCES

[1] Dabbagh, M., Kakavand, M., Tahir, M., & Amphawan, A. (2020, September). Performance analysis of blockchain platforms: Empirical evaluation of hyperledger fabric and ethereum. In 2020 IEEE 2nd International conference on artificial intelligence in engineering and technology (IICAIET) (pp. 1-6). IEEE.

[2] Rajeswari, T. R., Shareef, S. K., Khan, S., Venkatesh, N., Ali, A., & Devi, V. S. M. (2021, July). Generating and validating certificates using blockchain. In 2021 6th international conference on communication and electronics systems (ICCES) (pp. 1048-1052). IEEE.

[3] Kim, H., Jang, J., Park, S., & Lee, H. N. (2021). Error-correction code proof-of-work on Ethereum. IEEE Access, 9, 135942-135952.

[4] Xiao, Y., Zhang, N., Lou, W., & Hou, Y. T. (2020). A survey of distributed consensus protocols for blockchain networks. IEEE Communications Surveys & Tutorials, 22(2), 1432-1465.

[5] Nguyen, B. M., Dao, T. C., & Do, B. L. (2020). Towards a blockchain-based certificate authentication system in Vietnam. PeerJ Computer Science, 6, e266.

[6] Saleh, O.S., Ghazali, O., & Rana, M.E. (2020). BLOCKCHAIN BASED FRAMEWORK FOR EDUCATIONAL CERTIFICATES VERIFICATION.

[7] Yadav, A. S., Singh, N., & Kushwaha, D. S. (2023). Evolution of Blockchain and consensus mechanisms & its real-world applications. Multimedia Tools and Applications, 82(22), 34363-34408.. Evolution of Blockchain and consensus mechanisms & its real-world applications. Multimedia Tools and Applications, 82(22), 34363-34408.

[8] V. Buterin, "Ethereum white paper: a next generation smart contract & decentralized application platform," 2013, available at: http://www.theblockchain.com/docs/Ethereum_white_papera_next_generation_smart_contract_and_decentralized_application_platf orm-vitalik-buterin.pdf.

[9] Vujičić, D., Jagodić, D., & Ranđić, S. (2018, March). Blockchain technology, bitcoin, and Ethereum: A brief overview. In 2018 17th international symposium infoteh-jahorina (infoteh) (pp. 1-6). IEEE.

[10] Guo, H. and Yu, X., 2022. A survey on blockchain technology and its security. Blockchain: research and applications, 3(2), p.100067.

[11] A. Welligton dos Santos Abreu, E. F. Coutinho and C. Ilane Moreira Bezerra, "Performance Evaluation of Data Transactions in Blockchain," in IEEE Latin America Transactions, vol. 20, no. 3, pp. 409-416, March 2022, doi: 10.1109/TLA.2022.9667139.

[12] Fan, C., Ghaemi, S., Khazaei, H., & Musilek, P. (2020). Performance evaluation of blockchain systems: A systematic survey. *IEEE Access*, *8*, 126927-126950.

[13] B. Lashkari and P. Musilek. A comprehensive review of blockchain consensus mechanisms. IEEE Access, 9:43620–43652, 2021.

[14] M. M. Islam, M. M. Merlec and H. P. In.(2022). A Comparative Analysis of Proof-of-Authority Consensus Algorithms: Aura vs Clique. IEEE International Conference on Services Computing (SCC), Barcelona, Spain, 2022, pp. 327-332, doi: 10.1109/SCC55611.2022.00054.

[15] De Angelis, S. (2018). Assessing security and performances of consensus algorithms for permissioned blockchains. arXiv preprint arXiv:1805.03490.

[16] Zhang, X., Wang, Q., Li, R., & Wang, Q. (2022, May). Frontrunning block attack in poa clique: A case study. In 2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (pp. 1-3). IEEE.

[17] Zhang, Xinrui & Li, Rujia & Wang, Qin & Wang, Qi & Duan, Sisi. (2023). Time-manipulation Attack: Breaking Fairness against Proof of Authority Aura. 2076-2086. 10.1145/3543507.3583252.

[18] Hashim, F., Shuaib, K., & Sallabi, F. (2021, December). Performance Evaluation of Blockchain Consensus Algorithms for Electronic Health Record Sharing. In 2021 Global Congress on Electrical Engineering (GC-ElecEng) (pp. 136-143). IEEE.

[19] Islam, M. M., & In, H. P. (2023). Decentralized Global Copyright System Based on Consortium Blockchain with Proof of Authority. IEEE Access.

[20] M. M. Islam, M. M. Merlec and H. P. In.(2022). A Comparative Analysis of Proof-of-Authority Consensus Algorithms: Aura vs Clique. IEEE International Conference on Services Computing (SCC), Barcelona, Spain, 2022, pp. 327-332, doi: 10.1109/SCC55611.2022.00054.

[21] Kuperberg, M. (2020, August). Towards an analysis of network partitioning prevention for distributed ledgers and blockchains. In 2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS) (pp. 94-99). IEEE.

[22] Dhulavvagol, P. M., Bhajantri, V. H., & Totad, S. G. (2020). Blockchain ethereum clients performance analysis considering E-voting application. Procedia Computer Science, 167, 2506-2515.

[23] Rouhani, S., & Deters, R. (2017, November). Performance analysis of ethereum transactions in private blockchain. In 2017 8th IEEE international conference on software engineering and service science (ICSESS) (pp. 70-74). IEEE.

[24] De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., & Sassone, V. (2018). PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain. In CEUR workshop proceedings (Vol. 2058). CEUR-WS.

[25] Seri, P. R. (2021). Blockchain Based e-Voting. Southern Illinois University at Carbondale.

[26] Shi, E. (2019, June). Analysis of deterministic longest-chain protocols. In 2019 IEEE 32nd Computer Security Foundations Symposium (CSF) (pp. 122-12213). IEEE.

[27] De Angelis, S., Lombardi, F., Zanfino, G., Aniello, L., & Sassone, V. (2023). Security and dependability analysis of blockchain systems in partially synchronous networks with Byzantine faults. International Journal of Parallel, Emergent and Distributed Systems, 1-21.

[28] Hu, Y., Tian, G., Jiang, A., Liu, S., Wei, J., Wang, J., & Tan, S. (2023). A practical heartbeat-based defense scheme against cloning attacks in PoA blockchain. Computer Standards & Interfaces, 83, 103656.