Zero Knowledge Blockchain – A Zero Knowledge authentication system for applications that run on a Blockchain

Akhilesh Iyer
Cyber Security Center
Saint Peter's University
NJ, USA
aiyer@saintpeters.edu

Alberto La Cava
Cyber Security Center
Saint Peter's University
NJ, USA
alacava@saintpeters.edu

***Abstract*—Zero Knowledge Proof(ZKP) is a mathematical concept that allows an entity to prove its knowledge of something without actually revealing what it is. Such a concept is quite useful in the current scenario where authenticating users has become extremely difficult using existing defence systems like those built on the Public Key Infrastructure(PKI). This infrastructure has been successfully compromised over time using loopholes like single-point-of-failure(POF), corrupt 3rd parties, inefficient encryption, and many more. This project uses the concept of Zero Knowledge key exchange to design an authentication system that addresses the loopholes related to mathematical complexity. At the same time, to address architectural loopholes, the logic has been implemented on a blockchain network which together creates Zero Knowledge Blockchain: a system that combines the advantages of both ZKP and blockchain compensates their shortcomings and achieves faster, more secure authentication. The overall asymptotic complexity of traditional ZKP is generally O(nlogn) or more, where n is the size of the statement to be proved. With my solution, it is reduced to O((nlogn)/p) where p is the number of nodes in the blockchain that are involved in the process. The time taken to login with all nodes running on a single device is roughly 20s. If that is scaled for a real-life network with 7 active nodes, it turns out to be 3 seconds and the time lost in network lag. This is still much faster than the average login time for websites which is around 8 to 10 seconds.**

***Index Terms*—Zero Knowledge Proof; PKI; Authentication; Blockchain; Decentralized**

I. Introduction

Applications that run on conventional systems nowadays have become increasingly unsafe as attackers are discovering and inventing more effective and sophisticated ways to crack their security. Their attacks have grown from mathematical decipherment of encrypted data to finding loopholes in organizations, their staff, internal networks, ineffective access control, and even faulty protocols. Infrastructures like the PKI and network topologies like regular distributed systems that were considered to be state-of-the-art are being successfully attacked and compromised. These attacks target users' Confidentiality, Integrity and Availability (CIA) and tamper their access to software and data by targeting Authentication, Authorization and Accounting.

Authentication in-specific is very important to an application as it acts as a gateway filter that prevents unauthorized users from accessing anything that they don't have permission to. Therefore, compromising it can lead to subsequent compromises in other aspects of security as well.

Dong et al. have addresses issues with PKI in their paper [4] where they talk about topological complexity, unreliable 3rd parties, single Point-of-Failure where compromising a server or a group of servers stacked at a "single" location compromises all data, etc. While proposing a solution for them, they also discuss how PKI systems are susceptible to Man-in-the-Middle attack where the attacker appears to be a valid communicating party to both the sender and receiver, and to Distributed Denial of Service(DDoS) attacks which impede the availability of services or data to legitimate users.

A possible solution to these problems is using Zero Knowledge Proof, which is what this paper proposes. It is a way of proving that a user has some information without revealing what it is. An implementation of it is the Schnorr protocol [11] which exchanges credential information via a challenge-response mechanism where individual challenges are calculated as exponents of small primes to numbers derived from plaintext credentials. The responses are calculated in a similar manner user challenges as inputs.

Their verification happens easily due to the homomorphic property of multiplication. Breaking such an algorithm is extremely difficult as it involves multiplication of large primes and the discrete logarithm of very large numbers, both of which are computationally very costly. The working of this system is shown in Figure 1, where the variables have positive integer values and *exp()* is the exponentiation function. The implementation approach will be elaborated on, in the latter sections of the paper.

However, the computation required from the prover(user) and verifier(server) is also really high and is impractical if done on a conventional architecture, like the one on which PKI works. In their paper, Park et al. mention loopholes in PKI [10], like when an attacker compromises a user's device for its private key, or directly the Certification Authority(CA), the authority that issues certificates, generates and distributed key pairs. PKI

```
 A, B :            principal
 Na, Nb :          fresh number
 Sa :              private key
 Pa = exp(g,Sa) :  public key

A chooses Na and computes a = exp(g,Na)
 1.    A  ->  B   :     a
B chooses Nb
 2.    B  ->  A   :     Nb
A computes r = Na + Nb × Sa
 3.    A  ->  B   :     r
B checks that exp(g, r) = a × exp(Pa,Nb)
```

Figure 1. Schnorr ZKP key exchange

systems are decentralized as far as the protocol is concerned, but the computation is strictly local/centralized which creates the aforementioned vulnerabilities.
Iyer et al. proposed a solution to this in their paper [7] by implementing the protocol on a serverless architecture, thereby distributing the computational load among multiple ad-hoc servers that provide Functions-as-a-Service. Taking this as a basis, this paper proposes to implement ZKP on an application that runs on a blockchain network. It is a regular blockchain where nodes are incentivized to perform complex calculations like exponentiation, generate challenge-response pairs, and store credential data. The exchange of data that happens for the same is stored in the form of transaction hashes inside blocks which are mined and retrieved by whatever consensus protocol and retrieval mechanism the blockchain uses. This structure is very helpful as it ensures that data is stored almost permanently, load and responsibility are distributed, and there is no foul play due to the presence of incentives and peer-ranking; a system where nodes rank their neighbours based on parameters like response-time or integrity.
However, there are some issues with blockchain as well. As Mittal et al. describe in their paper [9], blockchains are extremely susceptible to Sybil attacks: where a user pretends to be multiple users at once and causes problems in the network. This is when users can attack consensus: a very basic principle of blockchain networks, which allows passing or failing of any new block or protocol change if more than 50% of the nodes vote for or against it. The attacker can therefore vote as more than one node and conduct the 51% attack where an attacker skews the network towards a decision by influencing more than 50% of the nodes to act in a particular manner. Other vulnerabilities include crypto-wallets, which are used to store crypto-currency like Bitcoin, Ether or AR, which the currency of the blockchain network Arweave [12]. Park et al. [10] point out the vulnerabilities of crypto-wallets that can be hacked by an unlocking script which compromises a node's private key and cryptocurrency.
To counter these ever-evolving attacks, stalwarts like IBM and Hyperledger have started developing systems like Indy which use ZKP for information security in their blockchains. Researchers like Yang et al. have also done commendable work on using ZKP for data privacy in their paper [13] where they use a combination of ZKP, homomorphic encryption, Pederson commitment and CL signature to reduce the complexity of private information retrieval to O(n).
This paper remediates these issues by combining the concepts of ZKP and blockchain to create Zero Knowledge Blockchain which is a mathematically impenetrable system that can authenticate users in real-time with very high accuracy and speed. It is different from the others in the manner of distribution of individual tasks and the use of Proof of Access [12] as its consensus algorithm which provides more robustness and flexibility.

II. LITERATURE AND RELATED WORK

This section summarizes the work that has been done on creating secure systems and categorized them according to relevance and coherence.

A. BLOCKCHAIN SECURITY

Park et al. [10] have discussed in their paper about how blockchain may be a secure manner to implement a distributed system, but it has many security risks in its basic structure. They implement it with PKI and observe that risks increase and efficiency decreases. They also propose that there are ways to remediate them, but they are not very effective if an attacker is computationally and politically strong.
Addressing another issue in distributed systems like blockchain network, Li et al. [9] talk about SybilControl, a mechanism that can curb Sybil attacks. These are attacks where a user masquerades as multiple users and targets a specific entity for either its information or money. They propose a solution where a node is identified by the number of computational problems it solves in a time period, given that it is resource bound. The node can then be traced and forbidden from attacking. However, assuming a node to be resource bound is not practical in the current scenario and therefore the solution is inapplicable.
These papers point towards the high level of architectural security and reliability that blockchain provides if implemented correctly.

B. ZERO KNOWLEDGE PROOF SECURITY

Iyer et al. [7] discuss about a system similar to the one described in this paper. They implement the Zero Knowledge key exchange on a serverless architecture, where each container server provides an individual Function-as-a-service and helps to achieve faster and more mathematically complex authentication. This paper presents a major improvement by changing the architecture to a blockchain, which mitigates problems like corrupt 3rd party servers, unreasonable costs of service, monopoly in the provision of services, easy identification of the location of servers resulting in their takedown, impermanence of data, difficult tracing of errors and attacks, etc.

Other solutions include the one by Hammond et al. [5], who discuss in their paper about modern methods of authenticating a user and its data and how they face many problems like brute-force cracking, dictionary attacks, implementation complexity, reduced possibility of reward, etc. They then show how ZKP solves these problems by a series of patented diagrams and flowcharts, along with its advantages and limitations.
Yang et al. [13] also address the ingenuity of ZKP in their paper where they debunk traditional anonymous password authentication techniques which have several performance weaknesses. They propose solutions that eliminate the need to save credentials and use anonymous passwords as passes instead. A pass is given to the server to ”retrieve” credentials. This makes the system fast, and the use of concepts like Homomorphic encryption, ZKP, Pederson commitment and CL signature makes it secure.
The use of ZKP in these papers and the results they get, prove that ZKP is a much better option for secure key exchange than regular PKI methods that are heavily dependent on vulnerable entities. It may be little more costly to implement than PKI, but the benefits outweigh its cost if implemented in a different environment.

III. THEORY BEHIND THE CODE

This section elaborates on the core theories behind Zero Knowledge Blockchain: Zero Knowledge Proof and Blockchain network.

A. *ZERO KNOWLEDGE PROOF*

After regular methods of authentication proved to be ineffective against sophisticated attacks, security enthusiasts turned to a technology called multi-factor authentication, where a pseudo-random One Time Password is generated after the regular username-password authentication and sent to the mobile number registered with the account via SMS. However, the National Security Agency deemed SMS as an insecure way of transferring data in 2017 and hence even this method of authentication doesn’t bring much relief now.
This is where ZKP came into the picture as it replaced the traditional system that uses key pairs and certificates with challenges and responses; i.e. the Prover-Verifier system. In this method, the client is a prover who openly declares its knowledge of a secret key to the server that is a verifier. It does so without actually revealing the source of knowledge about or the value of the secret key.
The verifier then begins its testing process by volleying challenges to the prover to validate its knowledge. The prover sends appropriate responses, as shown in Figure 1 to the verifier. That said, many popular algorithms have based their process on Zero Knowledge key exchange.
One of these is the Sigma Protocol [3], which implements a three-way challenge-response environment. Here, the prover first sends a message of declaration of its knowledge to the verifier, which responds by sending a challenge message back. The prover generates a response for it and sends it back to the verifier, which verifies it. The Schnorr protocol that this paper implements, is a specific use case of this Sigma Protocol.

B. *BLOCKCHAIN*

A blockchain is a data structure which is shaped and works like a linked list as shown in Figure 2 [8]. Data is stored in the form of their hashes inside virtual capsules or wrappers called blocks. These data are generally in the form of transactions that occur between different nodes in the blockchain network. Every block is linked to its previous block by its hash; i.e. the block hash of every block is seeded with the block hash of its previous block thereby creating an immutable link and relation between them. The process of creation of a block and its addition to the blockchain is called block mining. For a block to be successfully mined into the blockchain, it needs to get a vote of confidence from more than 50% of the nodes present in the network.
The algorithm that helps in this is called a consensus algorithm which is different for different networks. There are mainly 2 types of consensus algorithms: Proof of Work and Proof of Stake, the explanation to which is not required for this project but is given in these papers by Porat et al. [1] and King et al. [2]. There is also a newer consensus algorithm called Proof of Access which was presented by William et al. [12] in their paper, an adaptation of which has been used in this project.
The network itself is a distributed one, with peers acting as servers and clients. The data that is stored on the network is distributed between nodes according to the specification of the blockchain. The computation that happens to hash transactions and blocks is one of the core blockchain tasks, apart from storing the blockchain on a local device.
In order to run the blockchain network’s economy, there are incentive agents that provide monetary incentives in the form of cryptocurrencies to nodes that perform these core as well as secondary tasks. Due to these incentives, many nodes store their copies of the blockchain, and therefore there are very minimal chances of data being lost or getting corrupt. This also comes with the advantage that all transactions and relevant data are easily traceable and quickly retrievable.

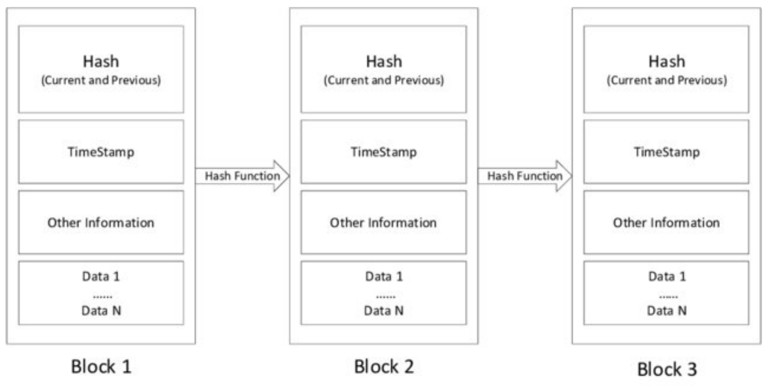


Figure 2. Structure of blockchain

IV. RESEARCH PROBLEM AND OBJECTIVES

The research problem that this paper mainly addresses is the inefficiency of current key exchange protocols and their inadequate security in settings that can create or find vulnerabilities in their architecture. The main focus among these protocols is the PKI system which suffers from multiple issues as mentioned in section 1 and 2.
To solve this problem, this paper focusses on achieving the following objectives:

1) To create and deploy the Zero Knowledge key exchange mechanism that was presented by Schnorr in his paper [11], as a real-time simulation in the presence of sensitive environment variables like network bandwidth, computational bottlenecks, race conditions, etc. and give it the form of an authentication system. The system makes use of a custom-made pseudo-random number generator to issue challenges and responses
2) To design and create a micro blockchain network and use it as the architecture to deploy the Zero Knowledge key exchange, with individual tasks in the entire process distributed among nodes in order to reduce computational load and time, and mitigate issues like a single point of failure, impermanence of data, untraceability of transactions, etc.
3) To integrate the aforementioned concepts in a manner that compensates both of their shortcomings and highlights their advantages in a way that improves both performance and security

V. WORK ENVIRONMENT

This section covers the hardware and software that was used to set the environment up for the simulation of Zero Knowledge Blockchain, the code for which is available on GitHub [6].

A. HARDWARE

- 16 GB RAM
- Intel® Core™ i7-8750H CPU @ 2.20GHz 2.20GHz
- 9 MB Cache
- 8 GB NVIDIA GeForce GTX 1060 GPU
- 1024 GB HDD + 256 GB SSD

B. SOFTWARE

- Windows 10 Home operating system (64-bit)
- Development and execution platform: JetBrains PyCharm Community Edition 2018.3.1 x64
- Development and execution environment:
- Python 3.7 with these mandatory libraries:
  - hashlib
  - _sha512
  - hmac
  - datetime
  - socket
  - pickle
  - math
  - functools
  - operator
  - pymysql.cursors
  - random
  - numpy
- Database software: MySQL Workbench 8.0 CE
- System firewall (Norton Security) with permissions to communicate via sockets on the localhost (127.0.0.1)

VI. IMPLEMENTATION

This section discusses the work methodology, the underlying process behind Zero Knowledge Blockchain, the data it works on, and the theoretical and performance benchmarks it meets.

A. WORK METHODOLOGY

To simulate user authentication on an application running on a blockchain network using the ZKP based Schnorr protocol, a program written in Python 3.7 will be executed on a GL63 MSI laptop with 16GB RAM and an i7 8th gen processor. There will be multiple Python files, 7 for now, each representing a node in the network.
These nodes will perform monetary and non-monetary transactions, serve as storage or function servers, mine blocks, and a lot more. A test application will be made to run on this network, which will either be coded using JavaScript or Python and will be just a login page. The logic behind the authentication will be the Schnorr protocol which is shown in Figure 1. This diagram is temporary and will be modified and improved as the project progresses.

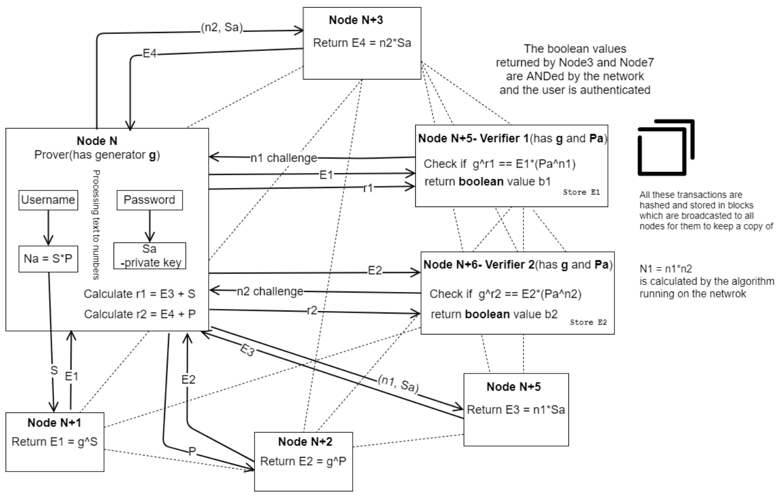

Node N+3
Return E4 = n2*Sa
(n2, Sa)
E4
The boolean values returned by Node3 and Node7 are ANDed by the network and the user is authenticated
Node N
Prover(has generator g)
Username
Password
Processing text to numbers
Na = S*P
Sa -private key
Calculate r1 = E3 + S
Calculate r2 = E4 + P
n1 challenge
E1
r1
Node N+5- Verifier 1(has g and Pa)
Check if g^r1 == E1*(Pa^n1)
return boolean value b1
Store E1
All these transactions are hashed and stored in blocks which are broadcasted to all nodes for them to keep a copy of
E2
n2 challenge
r2
Node N+6- Verifier 2(has g and Pa)
Check if g^r2 == E2*(Pa^n2)
return boolean value b2
Store E2
N1 = n1*n2
is calculated by the algorithm running on the netwrok
S
E1
E2
E3
(n1, Sa)
P
Node N+1
Return E1 = g^S
Node N+2
Return E2 = g^P
Node N+5
Return E3 = n1*Sa

Figure 3. The architecture of Zero Knowledge Blockchain

The credentials of a user will be accepted in plaintext format and processed using a generator function to get large numbers. Putting these values in the Schnorr protocol to generate challenges and responses will authenticate a user. The uniqueness of this project comes from the way these processes of computation and validation are organized.

As shown in Figure 3, the serverless architecture of the blockchain network will play a very important role as the processes to generate numbers via exponentiation will be performed as incentivized tasks by peer nodes. Even the validation processes will be performed by multiple nodes, thereby reducing the risk of single point-of-failure and irrecoverable loss of data. The nodes that perform these functions may or may not belong to the pool of nodes that are registered on that application.
The arguments/data that are being transferred amongst nodes will be stored as transactions on blocks, which will be appended to the blockchain after being verified with a consensus from more than 50% of the nodes in the blockchain.
The algorithm used for consensus will be Proof of Access(PoA) that was proposed by Williams et al. in their paper Arweave: A protocol for Economically Sustainable Information Permanence [12], which is built off of an existing algorithm, Proof of Work(PoW), but betters it in the computational overhead required.
Every block that has been successfully mined will be broadcasted to each node, which will store it in its SQL table that is connected to the node (in this case via an API) with the following columns:
$<$ **blockname, blockdata** $>$
blockdata will consist of the following:
$<$ **Block Hash, Timestamp, Nonce, Transaction ID, Transaction, Transaction Size** $>$
The block name will be the hash of the Transaction with a fixed secret string as its hash key.

The transactions are directly stored for now, instead of storing their hashes, for the purpose of clear demonstration. They will be broadcasted and stored only after being hashed, in the future. This ability of SQL to store data in a relational form will make it easy to maintain uniformity between tables and exchange data. This will, in turn, make the auditing, recovery and retrieval of blocks extremely easy and fast.
There are other and finer aspects to Blockchain like peer ranking, incentivizing different functions differently, content policies, etc. which are beyond the scope of this project and hence won't be implemented.

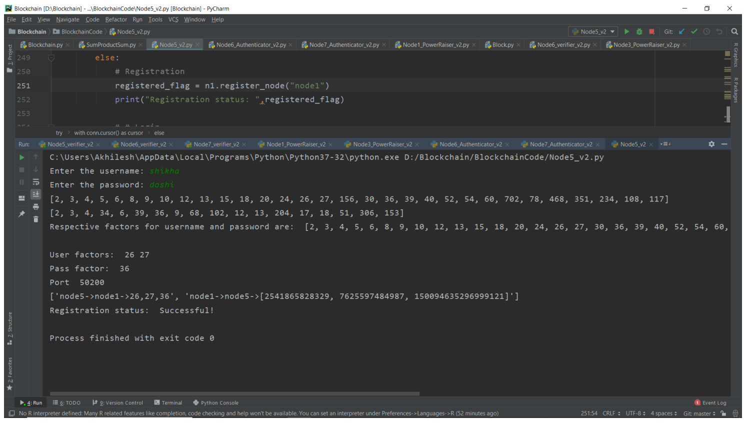
B. UNDERLYING PROCESSES

In the following section, there will be terminology which is defined as follows.
Each node can have four types of entities:

1) The *node* class, which perform core node functions like mining of blocks and is also the client that registers and logs in to the service
2) The *power_raiser* class, which takes exponent values from a client and returns powers of a generator value raised to those exponents. It may also mine a block with transactions if the client is seeking help with registration
3) The *authenticator* class, which is responsible for generating challenges, accepting responses and partially validating a client into the system
4) The *verifier* class, which is responsible for verification of blocks that are to be mined and added to the blockchain, by providing consensus

1) REGISTRATION:

- Open JetBrains PyCharm and let the libraries load. Open MySQL Workbench 8.0 and connect to the database server as a root user.
- Run all the verifier classes and any power_raiser class that you want the exponentiation service from. The verifier classes will keep running with sockets open until stopped manually.


Blockchain [D:\Blockchain] - ...\BlockchainCode\Node5_v2.py [Blockchain] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Blockchain BlockchainCode Node5_v2.py
Blockchain.py SumProductSum.py Node5_v2.py Node6_Authenticator_v2.py Node7_Authenticator_v2.py Node1_PowerRaiser_v2.py Block.py Node6_verifier_v2.py Node3_PowerRaiser_v2.py
try with conn.cursor() as cursor else
Run: Node5_verifier_v2 Node6_verifier_v2 Node7_verifier_v2 Node1_PowerRaiser_v2 Node3_PowerRaiser_v2 Node6_Authenticator_v2 Node7_Authenticator_v2 Node5_v2
4: Run 6: TODO 9: Version Control Terminal Python Console Event Log
251:54 CRLF UTF-8 4 spaces Git: master

```
        else:
            # Registration
            registered_flag = n1.register_node("node1")
            print("Registration status: ",registered_flag)
```

```
C:\Users\Akhilesh\AppData\Local\Programs\Python\Python37-32\python.exe D:/Blockchain/BlockchainCode/Node5_v2.py
Enter the username: shikha
Enter the password: doshi
[2, 3, 4, 5, 6, 8, 9, 10, 12, 13, 15, 18, 20, 24, 26, 27, 156, 30, 36, 39, 40, 52, 54, 60, 702, 78, 468, 351, 234, 108, 117]
[2, 3, 4, 34, 6, 39, 36, 9, 68, 102, 12, 13, 204, 17, 18, 51, 306, 153]
Respective factors for username and password are:  [2, 3, 4, 5, 6, 8, 9, 10, 12, 13, 15, 18, 20, 24, 26, 27, 30, 36, 39, 40, 52, 54, 60,

User factors:  26 27
Pass factor:  36
Port  50200
['node5->node1->26,27,36', 'node1->node5->[2541865828329, 7625597484987, 150094635296999121]']
Registration status:  Successful!

Process finished with exit code 0
```

No R interpreter defined: Many R related features like completion, code checking and help won't be available. You can set an interpreter under Preferences->Languages->R (52 minutes ago)

Figure 4. Entering credentials for a successful registration

- As shown in Figure 4, run any node class. If it is the first time that any node class is run, a genesis block will be created which will be verified by all other verifiers and inserted into the blockchain (their copies of the blockchain as SQL tables)
- If it is not the first time, the code will ask for the username and password. Once entered, they will be converted into numbers by the SumProductSum class which will be sent to the power_raiser class as exponents.
- It will perform long number exponentiation of a pre-shared generator to those exponents and return the raised powers. It will also broadcast the transactions wrapped in a block over the blockchain for verification, which will be added to the blockchain if there is consensus.

C. LOGIN

- Run all the verifier classes, two power_raiser classes that want you the exponentiation service from and two authenticator classes that you want the authentication service from. The verifier classes will keep running with sockets open until stopped manually.
- Run any node class with the login option. The code will ask for the username and password. Once entered, they will be converted into numbers by the SumProductSum class which will be sent to the power_raiser classes as exponents.

- They will perform long number exponentiation of a pre-shared generator to those exponents and return the raised powers.
- The powers are sent to the authenticator classes that generate challenges from them and send them to the client.
- The client calculates the appropriate responses and returns those values to the authenticators.
- They then fetch values from the transactions that are stored in their copy of the blockchain, process them, compare the processed values to the responses, and return True or False accordingly.
- If both the authenticators return True, the client is successfully logged in.
- There is however no block mined for login currently, as transactions are not hashed (for demonstration purposes) and can thus become duplicates of transactions in the registration blocks which could in turn pose problems for authenticators

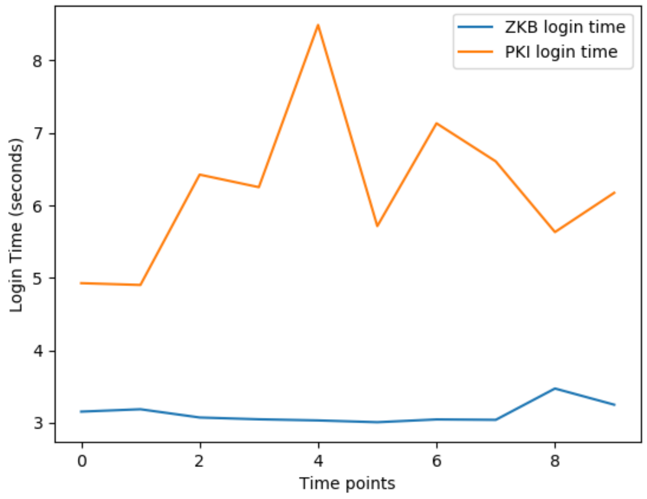


Figure 5. Graph comparing time required to login using ZKB (Blue) vs. time required to login using PKI (Orange)

VII. CORE RESEARCH RESULTS AND ANALYSIS

When tested under normal conditions in an environment mentioned in section 5, the project yields the following results:

1) The asymptotic complexity of the traditional ZKP key exchange process should be ***O(nlogn)*** or more, where n is the size of the statement that is proven. However, due to the uniqueness of its implementation in Zero Knowledge Blockchain, the complexity gets reduced to ***O((nlogn)/p)***, where p is the number of nodes involved in the login process.
2) The time taken for a successful login to a service that uses the PKI is ***10 seconds***. The average time required for logging into the service that runs on Zero Knowledge Blockchain is 22 seconds if all the 7 nodes participating in the exchange run on one machine, the one specified in section 5.1. Therefore, it is safe to say that if one node is run on one machine, the average time for logging in is around ***5 seconds***, which is almost ***2 times faster***. The difference in performance is evident from the graph shown in Figure 5. which compares the average time required to log into a ZKB system vs that required to login to a PKI system, both in ideal case scenarios.
3) There is the added complexity of an attacker finding it almost impossible to break the whole system due to the mathematical complexity of the discrete logarithm problem, which is defined as follows:
   Problem: *Let G be a group. Assume its group operation to be multiplication and identity element as 1. Let b be a member of G. Let a also be a member of G. Then, any integer k that solves the equation bk = a is called a discrete logarithm.*
   The problem is that there is no computationally efficient mathematical algorithm that can solve for k. There is also the added frustration for an attacker to identify each anonymous node performing an individual task from an unknown location and putting together all the pieces to make sense.

VIII. CONCLUSIONS

The conclusions that can be made from this project are:

- The authentication system is very simple in its core fundamentals but very effective and complex after implementation, due to both the text to number converter and the use of exponentiation in intermediate steps
- Because of consensus between nodes, there is a sense of democracy in the network which encourages pro-social and mutually beneficial behaviour between nodes. At the same time, this may also encourage a group of nodes to attack a new or existing node, and never vote positively for it
- The current system is also outstanding in that each function that is performed by a node is quite independent of the one performed by the others. This will mean more operational flexibility and higher performance for a node, and as there are no dependencies, there won't be any causal issues or chain vulnerabilities. This will also mean more robustness for the network, as there will be a very high probability that any node will have at least one function running at a given point in time.
  To add to it, the physical location and identity of each node is also (or will also be) hidden and different. Therefore, an attacker wanting to attack the whole system will need to attack almost the entire network, for which it will need infinite resources and is hence impossible. (There are 7 nodes in just the simulation. There may be more in an actual network)
- If each node has different servers/devices for each of the three functions mentioned in section 6.2, hacking it will become even more difficult.

- The attacks that trouble blockchain, like the 51% attack and the Man-in-the-Middle won't achieve much success here.
  The first one, because login and hashing power are mutually independent. As one node can get only one function at a time, more hashing power won't mean more reward.
  The second one, because of the futility of and difficulty for the attacker to find every serving node, sit in between it and client node, and try to find a sensible pattern that can give it any useful information.

IX. Future Work

Future work in this project may include the implementation of better consensus algorithms, the hashing of all transactions and mining of all blocks (even the ones containing login transactions) and setting up Raspberry Pi servers as nodes and performing real-life blockchain and ZKB functions on them. There could also be work on better block structures so that the requirement of storage is reduced. To improve it further, the blockchain could be arranged in a tree-like structure with transactions of similar type being batch processed for easier and unambiguous access.

References

[1] Pratap A. Shah P. Adkar V. 13. Porat, A. Blockchain consensus: An analysis of proof-of-work and its applications.

[2] Nadal S. 14. King, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper*. August 2012.

[3] I. Damgård. On -protocols. 2002.

[4] Chen Y. Fan J. Bai J. Zhang P. Li F. Dong, G. Anonymous cross-domain authentication scheme for medical pki system. *In Proceedings of the ACM Turing Celebration Conference-China*. pages 1–7, May 2019.

[5] Carlander S.J. Ricotta F.J. Hammond, I.F.J. *U.S. Patent No. 7,840,806*, washington, dc: U.s. patent and trademark office. 2010.

[6] A. Iyer. Iyerakhilesh/zeroknowledgeblockchain: Using zero knowledge proof algorithms for authentication on services that run on blockchain: https://github.com/IyerAkhilesh/ZeroKnowledgeBlockchain. 2018.

[7] Nagrikar S. Iyer, A. User authentication and identification via zero knowledge proof on serverless system. 2018.

[8] Han M. Wang Y. Joshi, A. A survey on security and privacy issues of blockchain technology. January 2018.

[9] Mittal P. Caesar M. Borisov N. Li, F. Sybilcontrol: Practical sybil defense with computational puzzles. *In Proceedings of the seventh ACM workshop on Scalable trusted computing*. pages 67–78, October 2012.

[10] Park J.H. Park, J.H. Blockchain security in cloud computing: Use cases, challenges, and solutions. 2017.

[11] Cramer R. Damgard I. Schoenmakers B. Schnorr, C.P. The schnorr protocol. 1991.

[12] Diordiiev V. Berman L. Uemlianin I. Williams, S. Arweave: A protocol for economically sustainable information permanence. 2017.

[13] Zhou J. Weng J. Bao F. Yang, Y. A new approach for anonymous password authentication. *In 2009 Annual Computer Security Applications Conference*. pages 199–208, IEEE, December 2009.