Private Blockchain for Visitor Authentication and Access Control

Ka Ying Chan
Mathematics and Computer Science
SUNY Oswego
Oswego, NY, USA
kchan2@oswego.edu

Mason Lovett
Department of Computer Science & Software Engineering
Butler University
Indianapolis, IN, USA
mlovett@butler.edu

Yesem Kurt Peker
TSYS School of Computer Science
Columbus State Univerity
Columbus, GA , USA
peker_yesem@columbusstate.edu

***Abstract*— This study applies the high data integrity that comes with blockchain technology towards authentication and access control for visitors of a physical facility. The use of smart contracts on an Ethereum based implementation of the blockchain allows for smart contract code to handle both access control and visitor authentication at scale. Javascript code executed off the blockchain enables the system to interact with and parse through the blockchain data. The proposed system is scalable, applies to multiple use cases, and mitigates issues a centralized approach faces.**

Keywords—blockchain, authentication, access control

I. INTRODUCTION

Blockchain is an emerging technology that provides a decentralized digital ledger. Data is grouped together and stored in chunks called blocks. Every time a new block is added, it is cryptographically attached to the previous block, creating a long chain of blocks that are all linked to one another. This means that changing the original block invalidates every single block afterwards, making the blockchain tamper evident. Blockchain is also decentralized which means that multiple computers called nodes are all connected to one another forming a blockchain network. Each node on the network participates in deciding what data is valid and can be added to the blockchain. Each node also stores a local copy of the data.

These features along with some others can help alleviate the issues that many centralized systems face. One such issue is a central point of failure. In a centralized system, a compromised central machine means the whole system is compromised. With blockchain this is not the case. Blockchain provides an environment where trust between nodes on the network is not necessary. As long as the rules of the network are defined properly, the nodes do not need to have trust in one another. In other words, the network can still function if some of the nodes are acting maliciously or are not responding at all. Another issue that traditional centralized systems can face is low data integrity. If data is stored in a centralized database, modifications can be made to old data. If the system is compromised, this means that data could be altered and no one would be able to tell. Blockchain is tamper evident which alleviates this issue.

This research was supported by the U.S. National Science Foundation under grant award # 195041.

For this project, the main benefit of blockchain that is leveraged is the immutability. Data is cemented into the ledger once a block is sealed or approved by the nodes. From this point on, that data only gets more and more difficult to modify as more blocks are linked onto it and appended to the ledger. This feature of blockchain technology allows the users to have a high level of trust on the integrity of the data. This project attempts to create a visitor authentication system which doesn't suffer from a centralized point of failure. The proposed system can tolerate a certain amount of unresponsive or maliciously acting nodes on the network and still properly track and authenticate users. This system improves upon a traditional centralized visitor authentication and access control system because it is harder to undermine and the data has a higher level of integrity.

II. BACKGROUND

Blockchain technology is a way of increasing the integrity of stored data. Blockchain works by segmenting data into groups of transactions called blocks. These blocks are cryptographically linked together, meaning that changing one block would change all other blocks added after it. This creates a system that is tamper evident, making it much more difficult to modify data than in a traditional database or central server. Blockchain technology relies on many different machines known as nodes all working together to form a network. Each node has a copy of the data that is being stored. This creates many copies of the data which are spread out over the network. Because of this, the system can tolerate a certain amount of nodes going down or even acting maliciously without the system failing to maintain data integrity. Data is added to the blockchain via data transfers called transactions. A transaction is sent to the network, and then the nodes decide whether it should be added to the official ledger or not. This process of deciding whether or not to add the contents of a transaction is called a consensus algorithm and there are multiple. Consensus algorithms make it such that a bad actor on one node cannot overtake the network easily. To overtake a blockchain network, an attacker would need to control the majority of the nodes. This makes attacking a blockchain system much more costly than attacking a centralized system.

There are three main categories of blockchains. The type which aligns most closely with the central philosophy of

decentralization is a public blockchain. A public blockchain is open for anyone with an internet connection to join. This allows for the network to grow very large and thus makes it difficult to attack. At this point it is worth noting that blockchain technology provides high data integrity but since every node has access to the data, it does not guarantee confidentiality of the data. Because of this public blockchains are not always appropriate. Another type of blockchain is a consortium blockchain. A consortium blockchain is used between a set number of entities. Unlike a public blockchain, not just anyone can join. Typically consortium blockchains are used for businesses or associations. These different entities will work together to have more nodes in the network than they could individually. This is mutually beneficial as each node makes the blockchain harder to attack. The final type of blockchain is a private blockchain. Private blockchains are blockchains where a single entity or business controls the network for personal use. This can be advantageous because the entity in charge has much more control over the network. This can make processing transactions more efficient and reduce the overall storage required for the data.

The blockchain used in this study is Ethereum. Ethereum maintains one of the largest cryptocurrency blockchains as of this time second only to Bitcoin. The native cryptocurrency on Ethereum is called Ether. Ethereum is open source and can be replicated by entities to create their own blockchain networks. This means Ethereum isn't strictly used for cryptocurrency and can be implemented or optimized for other uses of blockchain technology. Ethereum allows for execution of code on the blockchain via pieces of code called smart contracts. Smart contracts can perform logic on the data that is received from transactions and can conditionally emit messages called events that are stored on the blockchain. These events are how programs running outside of the blockchain network can gather information about what is going on within the blockchain. However events take up storage space on the blockchain and can cause the size of the blockchain to inflate quicker over time. Solidity is a common language for writing smart contracts.

III. RELATED WORK

Blockchain technology has found applications in various areas. Yaga et al. describe and detail the different aspects, use cases, qualities, and models of blockchain technology and other inherited processes like consensus models and cryptographic hashes [1]. Lesavre et al. apply blockchain technology for identity management and discuss authentication, authorizations and data sharing within organizations or on the web [2]. Hammi et al. propose a system which attempts to solve certain security issues in IoT devices using Blockchain technology [3]. They propose a framework, called Bubbles of Trust, where trusted devices are grouped into a bubble. One device in the bubble is chosen to be the master and is responsible for signing all other devices in the bubble. All devices in the same bubble have some identifiers and signatures that can prove their identities in the bubble. Only devices in the same bubble can communicate with each other. This framework can be further developed such that cross-bubble communications are allowed between chosen bubbles, and devices can be revoked if they are compromised. In [4], Bouras et al. aim to produce an efficient, scalable, decentralized IoT identity management system with low energy consumption. Their model is based on Hyperledger Fabric and uses a permissioned consortium blockchain. Every node on the network must have a credential issued by a certificate authority who maintains the nodes in the system, authenticates interactions occurring in the network, and verifies the integrity of transactions by digital signatures. Registration, authentication, and revocation are split into three separate ledgers, each of which is managed by some organizations on the blockchain network. These ledgers can communicate using private channels. This model can be improved by automating the registration process and enabling multi-factor authentication. Ouaddah et al. attempt to build a secure access control system while letting users have more control over their information [5]. The proposed system, called FairAccess, categorizes all parties with public/private key pairs in the system into resource owners, requesters, and resources. Each user has a wallet that manages all the addresses. Resource owners can register resources, grant access, and revoke access. Requesters can request access and delegate access. When access is granted, tokens are given out along with the access control policies for the specific resources. Requirements stated in the policies have to be fulfilled to unlock the tokens. In this way, users do not have to expose their information to a third party in order to gain access to resources. In [6], Cha et al. propose a framework address the privacy issues in IoT devices. The framework includes the use of a blockchain gateway. All the device information and users' privacy preferences are handled by the gateway and stored on the blockchain. Users gain access to devices by signing their preferences to get tokens in return. Once tokens are validated by the devices, access is granted. This framework has drawbacks exposed in [7] where authors discuss attacks against the framework that lead to loss of privacy and trust. Yavari et al. claim that Cha et al.'s attempt is vulnerable to secret disclosure attack, replay attack, traceability attack, and reuse token attack and propose a new protocol called IBCbAP is to eliminate the vulnerabilities. Modifications made include (1) using a new signature scheme (2) adding a new random value to each session (3) adding nonces and timestamps to messages (4) using random numbers generated by all parties in the network in tokens. There also have been proposals for using blockchain and smart contracts for role-based access control. RBAC-SC makes use of Ethereum's smart contract technology to realize a trans-organizational utilization of roles [8]. Authors use the blockchain technology and smart contracts as the trust and endorsement relations that are essential in the RBAC and utilize a challenge-response protocol for authentication. In [9] authors propose a RBAC model where three entities, namely role issuer, role owner, and role verifier, are considered. A role owner, an externally owned account, is a user whose role is issued by the role issuer. The role verifier sets up the required access for the services and authenticates the role owner. In [10] a similar model is proposed with similar entities: role issuer, resource owner, and user. The proposed model uses a blockchain-based smart contract for managing user-role permissions in an organization.

IV. Proposed Work

In this study we propose the use of a private blockchain for achieving somewhat continuous authentication and access control of visitors on a site. We use the term somewhat continuous to emphasize that a visitor is authenticated to the system at regular time intervals and these intervals can be shortened to get close to a continuous authentication system. We consider the site to be divided into sections and that a visitor has access to only some of the sections. We assume that the site has access points (APs) that are distributed over the site. The site also provides hardware tokens to visitors with these characteristics:

- Has a mechanism to detect where a visitor is on the site (e.g. GPS),
- Has a biometric scanner (e.g. fingerprint scanner),
- Has capability to submit transactions to the blockchain

Our proposed system has three main components as is described below and depicted in Fig.1.

1) Visitor management

Visitor management deals with visitor registration and revocation. It includes a server that is responsible for getting visitor authentication and access information (i.e. disallowed locations) and putting it on the blockchain via a smart contract as well as disabling the smart contract when necessary. The server also stores the information on a token. The token can be an app on a smart device or a physical dongle that the visitor has to carry with them. The authentication information can come from a fingerprint or another biometric based on the capabilities of the token. Allowing different types of tokens will ensure accessibility in the system.

2) Visitor reporting

Visitor reporting deals with visitor re-authentication and updates on the location of the visitor. As the visitor is moving around on the site, they rescan their fingerprints using the token. The token also gets the location information and sends (reports) the fingerprint scan along with the location, to the blockchain at the address of the smart contract created for the token during visitor registration. The tokens connect to the blockchain network through one of the approved access points that is on their list.

3) Visitor monitoring

Visitor monitoring is about detecting abnormal behavior(s) as the visitors are on the site. Visitor monitoring starts on the blockchain and goes on at the trusted monitoring server. The smart contract that is created for the token when the visitor registered has these functionalities:

- Check if the length of time since the visitor token last reported is more than a preset value,
- Compare the newly submitted hash of fingerprint with the one that was stored when the smart contract was created,
- Check if the location of the visitor is in the set of disallowed sections,

The smart contract emits an event when one of these incidents happens:

- The visitor has not reported to the chain for more than a preset amount of time.
- The fingerprint hash does not match the initial fingerprint hash stored for the user
- The visitor is in a disallowed section

An event specifies which of the above abnormal cases has occurred. The events are picked up by an off-chain server. The off-chain server is a trusted server that listens to the blockchain events and displays the information about abnormal cases. If desired, this server can be used for further action such as requesting the visitor management server for revocation of the visitor's smart contract or locking entries for the visitor, etc.

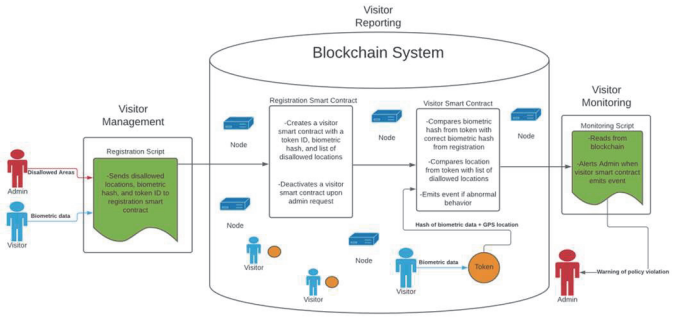

Fig. 1. Overview of Proposed System

B. Implementation

We implement our proposed system on a private Ethereum blockchain. We simulate the tokens in software; that is, we generate a random hash value for fingerprint hashes, a random set of disallowed sections for the visitors and ids for the tokens. We use a laptop with a Windows 64-bit operating system, an Intel(R) Core(TM) i5-1035G1 CPU x64-based 4-core processor with base speed of 1.19 GHz, 8 threads, 20 GB RAM, and 477 GB SSD; and a desktop computer with a 6-core 3.6 GHz CPU, 16 GB of DDR4 3200Hz RAM, and a discrete graphics processing unit running an Ubuntu virtual machine with 12 GB of dedicated RAM and 4 dedicated cores for the visitor management and monitoring servers and Raspberry Pi's (Model 4) for access points (APs). All the APs and the servers are nodes on the blockchain. The tokens act as wallets on the blockchain; they do not participate in mining or the consensus algorithm. The visitor management and monitoring servers are trusted laptops. The access points are also trusted in the sense that their setups are done by trusted users of the system. The untrusted elements in the system are visitors and any person or process that attempts to inject false data into the blockchain or interfere with the visits.

In this section we elaborate on how the different components of the system work. We provide step by step descriptions of each component.

1) Visitor Management

The main component of visitor management is the visitor registration process. The visitor management also deals with visitor revocation.

Visitor registration: The registration process starts on a trusted server which we call the "visitor management server" and is completed on the blockchain. There are two smart contracts in the registration process. The first is a universal contract which is in charge of registering the visitors. We will call this contract the "registration contract". The registration contract receives the authentication and access information (disallowed sections) of the visitor from the visitor management server along with a token id and creates a brand-new contract for the visitor. This newly created contract is given a new address and will be referred to as the "visitor contract". The address of the visitor contract is provided to the token that the visitor ought to be carrying with them. The token now can send the biometric hash and the location information to the visitor contract address at predefined intervals. At this point the registration process is complete. A sequence diagram for the registration process is given in Fig 2. Pseudocodes for User Registration Contract and Visitor Contract are given in Pseudocodes 1 and 2, respectively.

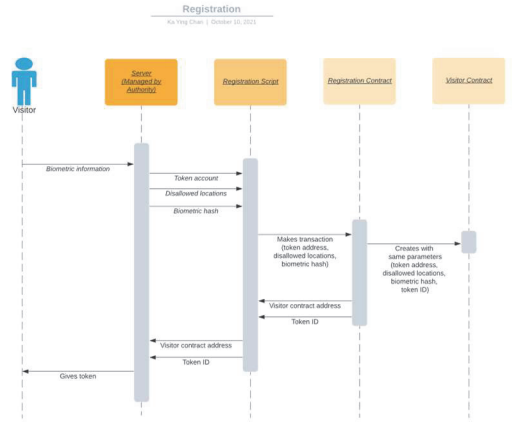

Fig. 2. Registration

Pseudocode 1. User Registration

```
function registerVisitor(visitorBiometric, disallowedLocations) {
   tokenAccount = server.generateAccount();
   biometricHash = SHA3Hash(visitorBiometric);
   transaction = sendTransaction({from: server; to: registrationContract}
                 tokenAccount, biometricHash, disallowedLocations);
   return Token(tokenAccount, transaction.visitorContractAddress);
  }
```

Pseudocode 2. Visitor Contract

```
function startToken(){
locationTimer = new Timer(1);     // 1 minute
biometricTimer = new Timer(5);     // 5 minutes
while (token.isActive()) {
      if (locationTimer.timeIsUp()) {
         location = token.getCurrentLocation();
         sendTransaction({from: tokenAccount; to: visitorContractAddress}, location)
         }
      if (biometricTimer.timeIsUp()) {
          biometric = token.askForBiometric();
          sendTransaction({from: tokenAccount; to: visitorContractAddress}, location
          }
      }
  }
```

In the implementation, the registration contract is designed so that only information from the visitor management server is accepted. This avoids unauthorized external devices from creating contracts.

Visitor Revocation: When a visitor ends their visit and exits the site or if there is a problem with the visit communicated by the monitoring server, the visitor management system revokes the visitor contract by disabling it.

2) Visitor Reporting

Visitor reporting is implemented via the visitor smart contract that was created for the visitor during registration. The smart contract expects an update from the visitor's token at regular time intervals. There are two updates: One for biometric hash and one for location. In our implementation the length of the interval is set to 5 minutes for fingerprint hash and 1 minute for location. That is, the token has to report the location every minute and the visitor needs to scan their fingerprint every 5 minutes. These parameters can be adjusted based on the needs of the system. The smart contract is designed so that if the updates are not received in a timely manner, an event is emitted for monitoring purposes. This is explained in Visitor Monitoring below. Fig 3. presents how the smart contract works in a sequence diagram.

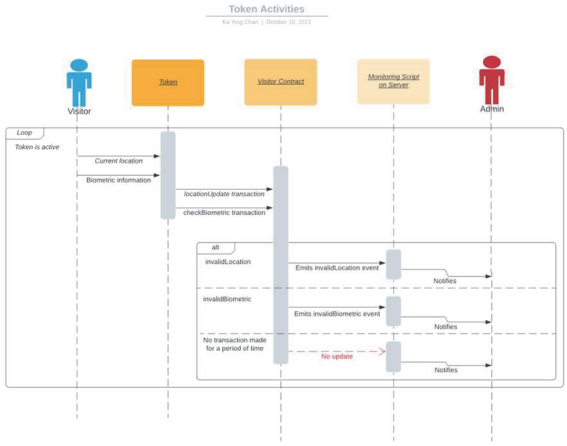

Fig. 3. Token Activities

3) Visitor Monitoring

Visitor monitoring is achieved by listening to the events emitted by the visitor contracts. The contracts are designed such that they only emit events that are considered abnormal. If a visitor is wandering around the site and updating at the expected times with no abnormal activity, then no event is emitted to outside the blockchain. This allows for a more efficient and less costly system in terms of space and time as less data need to be communicated to the outside world. Note that all transactions are still stored on the blockchain and can be accessed if needed.

The listening of the emitted events is done by a script running on the trusted "monitoring server" presented in Pseudocode 3.

Pseudocode 3. Monitoring Script

```
function startMonitoring(ourPrivateBlockchain)
   while (!end)
      latestBlock = ourPrivateBlockchain.getLatest();
      if (latestBlock.hasTransactions())
         for (Transaction t : transactions)
            eventLogs = t.logs;
            if (logs.isNotEmpty())
               event = logs[0].topics;
               if (event == tokenCreated)
                  output("New token created");
               else if (event == invalidLocation)
                  output("Token in invalid location");
               else if (event == invalidBiometric)
                  output("Token received invalid biometric")
               else
                  output("Unknown event");
```

In figures 4 to 6 below we include screenshots presenting how the registration, reporting, and monitoring steps work. Fig 4. shows a screenshot of a sample of running the registration script. At the end of the script,10 visitor contracts with tokens ids 0 to 9 are created. Fig 5. shows the location updates from the visitors. These updates are displayed for demonstration purposes. In actual use, the legitimate updates are stored on the blockchain; they don't need to be displayed. Fig 6. shows the monitoring script at work. The script displays the invalid locations and which tokens they are coming from. For example, in Fig 1. token 5 is registered with disallowed locations 54 and 27. In Fig. 5, it reports to be at location 27. Monitoring script captures the invalid location and displays token 5 in invalid location.

```
1634127170250 Creating new token with token number 5
1634127176923 account created
1634127176924 registering with locations 54,27
1634127177292 finished registration transaction
1634127177799 done registration
1634127177799 Creating new token with token number 6
1634127201570 account created
1634127201571 registering with locations 13,45,18,25,51,44,22,31,50
1634127209338 finished registration transaction
1634127209845 done registration
1634127209846 Creating new token with token number 7
1634127215485 account created
1634127215485 registering with locations 28,8
1634127221737 finished registration transaction
1634127222246 done registration
1634127222247 Creating new token with token number 8
1634127227604 account created
1634127227604 registering with locations 35,48,31,25,48
1634127230443 finished registration transaction
1634127230948 done registration
1634127230948 Creating new token with token number 9
1634127238486 account created
1634127238486 registering with locations 36,7,22,2,33,37,38,25,41
1634127238763 finished registration transaction
1634127239280 done registration
```

Fig. 4. Visitor registration on the visitor management server

```
1634127259127 Token 0 is sending location 26
1634127259129 Token 1 is sending location 11
1634127259131 Token 2 is sending location 13
1634127259131 Token 3 is sending location 4
1634127259132 Token 4 is sending location 23
1634127259133 Token 5 is sending location 27
1634127259134 Token 6 is sending location 11
1634127259135 Token 7 is sending location 29
1634127259136 Token 8 is sending location 51
1634127259138 Token 9 is sending location 16
```

Fig. 5. Location updates by visitors

```
1634127270022 Token 2 is in an invalid location. Current location: 13
1634127270033 Token 5 is in an invalid location. Current location: 27
1634127270042 Token 1 is in an invalid location. Current location: 11
```

Fig. 6. Inavlid locations captured on the monitoring script

C. *Results*

We analyzed our proposed system for storage and time requirements. For analysis, we simulated our implementation with 5 nodes and 10 visitors with each visitor expected to send location updates every minute and fingerprint hash every five minutes. These parameters can be changed to be more often or less frequent based on the use case.

We do not include a monetary cost analysis of running our system. Being a private network, there will be the usual costs associated with developing and maintaining the system. Unlike in public blockchain networks, the transactions used for registering visitors or reporting will not cost any fiat currency.

1) *Time Requirements*

The table below outlines the average timing from start to finish of the different processes present in our system. Registering a visitor in the system for the system consists of two processes: creating the token account and registering the user. Creating the token account entails telling the blockchain to create a new account for a token on the blockchain network, then giving this token account plenty of funds so that it can have enough gas to make transactions while it is active in the facility. Registering the user involves sending a transaction to the registration contract with the user's information and waiting for the registration contract to create a new visitor smart contract. Once these two registration processes are done, the sum of them makes up the total registration time.

The second central process timed was the alert notification timing. This process displays the average time between a token sending a transaction with a location violation and the monitoring system displaying the alert. The alert notification system consists of two processes. These two processes are the time it takes for the token to send a policy-violating update transaction and the time it takes for the system to display an alert to the administration. The update transaction row provides the time for either a biometric data authentication transaction or a location update transaction to be sent and then arrive at the user's corresponding visitor contract. The policy violation alert timing represents the amount of time between the abnormal transaction being published on the chain and the monitoring script to output an alert to the administration. The sum of the times for the transaction to be sent and the alert being output makes up the total time for the alert notification process.

Table 1. Average time for registration and alerts

	Average Time (milliseconds)
Token account creation	8907.9
User info registration	4532.8
Registration total	13440.7
Update transaction (location or fingerprint)	8471.0
Abnormal activity alert on the monitoring system (from when update goes on the blockchain to the alert)	1533.2
Alert notification total	10004.2

2) Space Requirements

After running the simulation for 10 minutes for 10 visitors with location updates every minute and fingerprint submissions every five minutes, we observed that the storage space required for the node that owns the chain (owner node) and the node that gets added as a peer (peer node) are different. After initializing the blockchain, the size of the "chaindata" folder of the owner node is 33,107 bytes and that of a peer node is 45,161 bytes. After running the simulation for 10 minutes, the size of the chaindata folder on the owner node is 276,147 bytes with an increase of 243,040 bytes, and that of a peer node is 288,194 bytes with an increase of 243,033 bytes. The difference between the amounts of increase is 7 bytes, which is negligible so we will assume that all nodes have the same size of chaindata. With this, we can estimate the storage requirement for longer periods. For example, the monthly storage for a site that accepts visitors 8 hours a day with 10 visitors each ten minutes can be estimated by

243,040 bytes/ten minutes * 6 ten minutes/hour * 8 hours/day * 30 days which is 349977600 bytes, approximately 0.35 GB.

D. Security Analysis

In this section we consider ways an adversary may want to compromise the system and the mechanisms we have in place to prevent these attacks:

1) Creating unauthorized visitor contracts

   Registration of users is done via the rregistration ccontract that lives on the blockchain. Attempts to create visitor contracts will not be honored if they are not initiated by the registration contract. In order to create a contract, the attacker will have to first compromise the trusted management server.

2) Impersonating a visitor

   In order for someone other than the visitor to do the reporting for the visitor's contract, they need to know the address of the contract on the blockchain. This information is stored on the token. If one steals the token, in order to report without raising suspicion, they would need the fingerprint hash of the visitor. The fingerprint hash is not stored on the token. It is on the blockchain and not visible outside the contract.

3) Tampering with the transactions

   An attacker may attempt to tamper with the transactions by changing the disallowed locations or fingerprint hash for a visitor. This can be done either via the registration contract or by changing the data on the blockchain. As explained in point 1 above, running the registration contract can't be done without compromising the trusted management server. And, changing the data on the chain is highly unlikely as the integrity of visitor reports is guaranteed by the tamper-proof property of the Blockchain technology

4) Unauthorized section in the time interval between updates

   A visitor has to regularly report to their smart contract. The visitor has to report both fingerprint hash and their location. The frequency of reporting can be set differently for the two types of reporting. The contracts are designed such that, if the visitor contract doesn't receive either of the reports on time, it emits an event indicating no location update or fingerprint hash is submitted. The frequency of the updates may be increased at expense of storage if necessary.

5) Tampering with the physical token

   Similar to point 4, the visitor contract will emit an alert if no updates are received from a token.

V. CONCLUSION

The results obtained indicate that this system is reasonable to implement in certain cases. The system could further be optimized per the use case by making modifications to the blockchain system, contracts, and scripts as necessary. The timing of the transactions could be modified as needed by implementing a different consensus protocol on the blockchain system. For this system, a proof-of-work consensus model was implemented. There are also proof of authority consensus protocols like the Clique proof of authority consensus protocol that would improve the transaction speed of the system. A new blockchain system could be developed specifically for this use case to further optimize the timing, and storage of this system.

The physical token could also be adjusted depending on the use case. A smart phone capable of gathering biometric data, gathering location information and interfacing with the access points in the system could be implemented.

The system could also be abstracted to monitor virtual visitors accessing virtual resources in some sort of file system or network.

ACKNOWLEDGMENT

We would like to thank Dr. Alfredo Perez, the PI of the REU Site at Columbus State University for the opportunity he provided us to work on this project.

REFERENCES

[1] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain Technology Overview," Gaithersburg, MD: National Institute of Standards and Technology (NIST), 2018.

[2] L. Lesavre, P. Varin, P. Mell, M. Davidson, and J. Shook, "A taxonomic approach to understanding emerging blockchain identity management systems," Gaithersburg, MD: National Institute of Standards and Technology (NIST), 2020.

[3] M. T. Hammi, B. Hammi, P. Bellot, and A. Serhrouchni, "Bubbles of trust: A decentralized blockchain-based authentication system for IOT," Computers & Security, vol. 78, pp. 126–142, 2018.

[4] M. A. Bouras, Q. Lu, S. Dhelim, and H. Ning, “A lightweight blockchain-based IOT identity management approach,” Future Internet, vol. 13, no. 2, p. 24, 2021.

[5] A. Ouaddah, A. Abou Elkalam, and A. Ait Ouahman, “FairAccess: A new blockchain-based access control framework for the internet of things,” Security and Communication Networks, vol. 9, no. 18, pp. 5943–5964, 2016.

[6] S. Cha, J. Chen, C. Su, and K. Yeh, “A blockchain connected gateway for BLE-based devices in the internet of things,” IEEE Access, vol. 6, pp. 24639–24649, 2018.

[7] M. Yavari, M. Safkhani, S. Kumari, S. Kumar, and C. Chen, “An improved blockchain-based Authentication Protocol for IOT Network Management,” Security and Communication Networks, vol. 2020, pp. 1–16, 2020.

[8] J. P. Cruz, Y. Kaji and N. Yanai, "RBAC-SC: Role-Based Access Control Using Smart Contract," in *IEEE Access*, vol. 6, pp. 12240-12251, 2018, doi: 10.1109/ACCESS.2018.2812844.

[9] Y. Lee and K. M. Lee, “Blockchain-based RBAC for user authentication with anonymity”, In Proceedings of the Conference on Research in Adaptive and Convergent Systems (RACS '19). Association for Computing Machinery, New York, NY, USA, 289–294, 2019, DOI:https://doi.org/10.1145/3338840.3355673

[10] P. Kamboj, S. Khare and S. Pal, “User authentication using Blockchain based smart contract in role-based access control” in Peer-to-Peer Netw. Appl. 14, 2961–2976 (2021). https://doi.org/10.1007/s12083-021-01150-1