CAHIERS MAGELLANES-NS
Volume 06 Issue 2
2024

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

USING BLOCKCHAIN FOR IOT ACCESS CONTROL AND AUTHENTICATION MANAGEMENT

Mrs. Misbah Kousar [1], Dr. Sanjay Kumar [2], Dr. Mohammed Abdul Bari [3]
[1]Ph.DScholar in Kalinga University
[2]Associate Professor, Kalinga University (CSE DEPT)
[3]Associate Professor –CSE-KMEC

Abstract: IoT devices are resource-constrained in terms of processing, storage, and networking capacity, which makes it a difficult task to secure access to them. Today's authentication and access control schemes exhibit substantial drawbacks as a result of their rapid dissemination and implementation. This paper suggests a blockchain-based solution that enables secure communication and authentication with IoT devices. Our solution capitalizes on the inherent characteristics of blockchain technology and augments existing authentication protocols. Specifically, our proposed blockchain-based solution, architecture, and design enable accountability, integrity, and traceability through tamper-proof logs. The paper offers a comprehensive overview of the system's design and architecture, as well as specifics regarding the testing and implementation of a realistic scenario as a proof of concept.

Keywords: Internet-Of-Things, Blockchain, Smart contractAuthentication, Access control.

1. Introduction

The Internet-of-Things concept is a network of interconnected computing devices in a variety of fields and configurations that can be deployed globally. In order to provide a specific service, these devices are capable of communicating with other devices, gathering, sharing, and processing information [1]. The INTERNET will be connected to more than 50 billion devices by 2020, as per experts from CISCO, Ericsson, and other companies [2]. Currently, IoT devices are in operation in a variety of sectors, including personal accessories, medical equipment, and household appliances. In order to facilitate this functionality, these devices must possess specific attributes. They should be capable of communicating with other heterogeneous devices and operating on low energy. Additionally, they should be capable of maintaining a consistent connection with the back-end, if one exists, and be able to receive patches as needed.

Authentication and access control are essential concepts for the secure management of computer resources and networks. These concepts should be redefined in the context of IOT in accordance with the aforementioned characteristics.

The limited resources draw-back must be considered when developing techniques and access control policies. Similarly, classical access control methods, such as ABAC (Attribute-based access control) and RBAC (Role-based access control), have been demonstrated to be inflexible, unscalable, and difficult to upgrade in IOT conditions [6,13].

CAHIERS MAGELLANES-NS
Volume 06 Issue 2
2024

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

Furthermore, the centralized perception of authentication, which necessitates that all devices communicate with a specific entity, is a significant disadvantage. The construction of a system that is contingent upon a trusted third party necessitates the assumption of a TTP that is consistently authentic and accessible. This results in a bottleneck surrounding the trusted party, which in turn impacts availability. In the event that the centralized entity is compromised, the model also fails. In addition, the TTP has the ability to alter records without regard for accountability. Blockchain technology can be employed to address these drawbacks in the IOT design [3].

The technology that underpins Bitcoin is referred to as blockchain [4]. A growing chain of records can be used to define it. The blockchain is designed to inherit effective characteristics, such as the decentralization, tamper-proofing, and equal access to blocks of records by all nodes. This concept can be applied to any application that necessitates a trusted third party to verify records or transactions. The blockchain technology has enabled the replacement of the trusted third party with a transparent, untampered block of records that is accessible through a distributed form. Consequently, the trust is transferred from a single entity to a decentralized community of blockchain nodes.

The smart contract is a highly effective approach that employs the blockchain concept. In 1996, Smart-Contract was initially defined as a self-operating or self-executing program [5]. To facilitate the development of blockchain automated applications, this method was reintroduced in the Ethereum blockchain. Events and logs are features of the Ethereum blockchain. An event is a response (returned value) from the smart contract to the user interface that is interacting with it. The primary objective of utilizing events and logs is to facilitate communication between smart contracts and the programs that interact with them.

2. Literature Review

In this section, a variety of existing methods are presented to address the issue of authentication management and access control in IoT devices. In addition to their advantages and disadvantages, Section 2.1 will examine conventional methods of authentication and access control. The solutions presented in Sect. 2.2 are unique in that they all rely on blockchain technology as the foundation for their concepts.

2.1 IOT Authentication Traditional Models

A fundamental method is to directly authenticate with each device by utilizing a combination of (username, password). This method offers adequate access control, as the administrator (owner) specifies and stores the roles and permissions of each authenticated user on the device. Nevertheless, this approach is not scalable and generates an overhead due to the fact that the user must authenticate to each machine independently. Classic IoT devices, such as IP cameras and Internet-accessed home utilities, exhibit this technique [6].

A more sophisticated solution is to authenticate using single-sign-on protocols. For example, when oauth2 is implemented as an authentication method, users attempt to access a device by authenticating with a trusted oauth2 provider. This reputable third party may include Facebook, Google, and so forth. The trusted entity grants access if they successfully authenticate and possess the necessary permissions [7].

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

Authentication to the trusted entity is required to access resources that are managed by the same individual [8]. Initially, the device serves as the oauth2 client and transmits an authorization request to the user when the user attempts to access the IOT device resources. Subsequently, the user authorizes the client to communicate with the authorization server, which is the oauth2 provider. Then, the IOT device communicates with the oauth2 provider to verify the user's authorization to access its resources [9]. By authenticating to a single entity, the user can access multiple entities, thereby saving time and effort. In addition, the integration of such a solution is typically facilitated by the fact that the oauth2 provider is a reputable third party.

Conversely, the risk of a single point of failure threatening the availability of the proposed approach is exacerbated by the reliance on a centralized entity. Furthermore, if the user's account or the centralized entity is compromised, the entire system is impacted. Phishing is a critical attack vector that has a high success rate and has the potential to result in the failure of this model. Furthermore, spear phishing campaigns are becoming increasingly sophisticated, precise, and intensive in recent years, which has the potential to deceive even the most educated users [10]. In 2016, 76% of organizations reported being phished, as indicated by the Symantec Latest Intelligence Report for June 2017 [11].

The methods that have been discussed thus far provide a valid implementation for IoT authentication. Nevertheless, they are susceptible to certain deficiencies that may compromise their availability, performance, and scalability. The following methodologies have the potential to be implemented as an IoT authentication management and access control solution that employs blockchain technology.

2.2 Blockchain-Based Authentication Models

A challenge-response method was introduced by Auth0 to authenticate to a server via the Ethereum blockchain. The auth0 approach's drawback is the necessity of a third-party authentication server. It is a hybrid solution that integrates the centralization of a trusted third party with the decentralization of blockchain. The "Single Point of Failure" or "Single Point of Trust" issue will resurface with this methodology. The centralized server is crucial, as it is involved in 62.5% of the entire operation, according to Auth0. This undermines the advantages of blockchain technology by increasing dependence on a centralized entity [12].

Blockstack is introducing the concept of a new decentralized internet. Applications for storage and authentication are present on this network. In a manner comparable to public key infrastructure (PKI), the system employs a user's keypair to authenticate. Unique JSON Web Tokens are generated when users are authenticated accurately. The JWT enables access to all pre-authorized resources through a single authentication process that has been previously verified.

Unfortunately, the operation of blockstack is contingent upon the fulfillment of numerous prerequisites. The initial prerequisite for this system is the utilization of a block stack browser, as its objective is to transition to a newly decentralized network. If not, the user's machine should be installed with the blockstack application. Furthermore, the system implemented two additional layers on top of the blockchain: the peer network layer and the storage layer, which resulted in an increase in operational overhead. Lastly, the Blockchain Name System serves as a substitute for conventional

CAHIERS MAGELLANES-NS
Volume 06 Issue 2
2024

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

DNS in the context of entity interaction [13]. BNS is one of numerous innovative implementations that are designed to replace the traditional domain name system.

3. Proposed Mechanism

The paper proposes a blockchain-based solution that features a unique system architecture. It resolves the deficiencies of existing solutions. Additionally, it should be portable and capable of operating on any network with minimal dependencies, in contrast to blockstack. It is intended for IoT devices that are deficient in processing power. It also introduces the concept of OAuth implementation through a smart contract, which allows users to log in once and manage all authorized devices without the need to log in separately for each IoT device. Furthermore, the IoT devices have the capacity to operate smart contracts, thereby emerging as self-profiting devices.
There are numerous benefits to employing the Ethereum blockchain as a platform for this solution. Ethereum boasts a robust development framework that is already established. An incentive for miners to resolve challenge hashes. Additionally, the Ethereum light client protocol is capable of operating on IoT devices with limited memory and processing power, which is crucial for the proposed solution [14].

One-Time Authentication: Directly authenticate to the blockchain and access the resource using smart contract tokens. In this scenario, the user authenticates with the smart contract, which verifies their identity. Subsequently, the smart contract determines whether the user is authorized to access the resources. The authentication process is conducted in a separate phase. Upon successful completion, the user is able to interact with the device using their preferred method, such as ssh, http, or https. It is an efficient method of achieving decentralized authentication. Section 4 will address the assessment of this solution.

3.1 Assumptions

The subsequent assumptions must be considered in order to implement this solution:

- One or more IoT devices are owned by the user.
- The private key of the user is safeguarded, as the Ethereum keystore is not compromised.
- The user maintains an Ethereum account.
- Both the user and the IoT device are linked to the Ethereum blockchain.
- The smart contract will be deployed by the user.

Fully functional, the system is capable of challenging the final assumption. A centralized smart contract that authenticates users to their respective IoT devices can be established. Nevertheless, this paper's objective is to circumvent dependence on a central entity. It is more appropriate to request that users deploy their own smart contracts in order to achieve complete control over their own systems. This will establish a decentralized blockchain that is powered by decentralized smart contracts, with each user possessing their own smart contract.

3.2 System Architecture

CAHIERS MAGELLANES-NS
Volume 06 Issue 2
2024

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

The steps of the authentication process are illustrated in the message sequence diagram in Fig. 1. The Ethereum wallet address of the user is used to authenticate to the smart contract. The smart contract broadcasts an Access token and the sender's Ethereum address if the user is valid. The smart contract disseminates information to the user and the IoT device. The user creates a package that includes the Ethereum public key, user IP, access duration, and access token. This package is signed with the Ethereum private key and subsequently transmitted with the corresponding public key. If desired, the package may be encrypted. Nevertheless, it is not necessary for the protocol to function. In this situation, integrity is of paramount importance; consequently, the message is signed. The content of the package is verified by the IOT device upon its receipt. If successful, the device provides the user with access from the sender's IP for the specified duration. Alternatively, the request is terminated if any of the aforementioned checks fails.

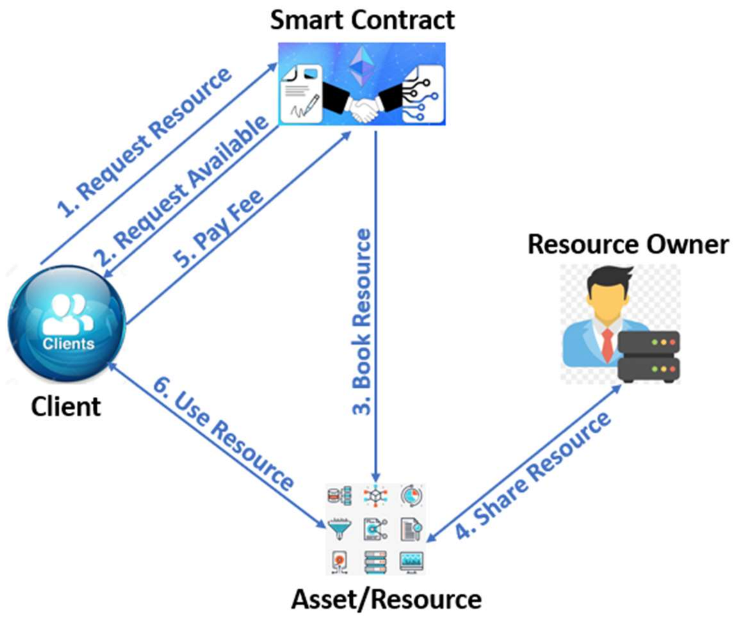

Figure 1: Proposed Architecture

The initial step of the smart contract involves the completion of its authentication process. It is acknowledged that the IOT device must execute numerous verification procedures. Nevertheless, this solution functions flawlessly on a standard Raspberry Pi 3 Model B. Working Model of Proposed model is described as follows:

<u>*Proposed Mechanism:*</u>

Step 1: Authorization Request

$AuthReq(I\rightarrow U){:}I$ requests authorization from U

$Req_{IU}=Request(I,U)$

Where Req_{IU} is the authorization request from I to U.

Step 2: Authorization Grant

$AuthGrant(U\rightarrow I){:}U$ grants authorization to I

$Grant_{UI}=Grant(U,I)$

Where $Grant_{UI}$ is the authorization grant from U to I.

CAHIERS MAGELLANES-NS
Volume 06 Issue 2
2024

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

Step 3: Forward Authorization Grant
$AuthGrant(I \rightarrow A):I$ forwards the authorization grant to A $Grant_{IA}=ForwardGrant(I,A,Grant_{UI})$ Where $Grant_{IA}$ is the forwarded authorization grant from I to A.
Step 4: Access Token Generation
$AccessToken(A \rightarrow I):A$ generates an access token and sends it to I
$Token_{AI}=GenerateToken(A,I,Grant_{IA})$
Where $Token_{IA}$ is the access token generated by A for I.
Step 5: Send Access Token to Resource Server
$AccessToken(I \rightarrow R):I$ sends the access token to R
$Token_{IR}=SendToken(I,R,Token_{AI})$
Where $Token_{IR}$ is the access token sent from I to R.
Step 6: Access Protected Resource
$ProtectedResource(R \rightarrow I):R$ validates the token and provides access to the protected resource to I
$Resource_{RI}=AccessResource(R,I,Token_{IR})$
Where $Resource_{RI}$ is the protected resource accessed by I from R.

The proposed mechanism outlines a detailed process for securing data transmission in an online social network environment using a combination of authorization requests, grants, and access tokens. Initially, the Internet of Things (IoT) device (Client), denoted as *I*, initiates the process by sending an authorization request to the User (Resource Owner), labeled as *U*. This step is represented by the equation $AuthReq(I \rightarrow U)$, where $Req_{IU}=Request(I,U)$. The User, upon receiving this request, evaluates the credentials and the need for access before issuing an authorization grant back to the IoT device, captured by $AuthGrant(U \rightarrow I$ and mathematically expressed as $Grant_{UI}=Grant(U,I)$. This grant signifies that the User has validated the IoT device's request and permits it to proceed to the next step.
Following the receipt of the authorization grant from the User, the IoT device forwards this grant to the Authorization Server, denoted as A. This is crucial for ensuring that the server can generate the necessary access tokens for secure data transmission, depicted by $AuthGrant(I \rightarrow A)$ and $Grant_{IA}=ForwardGrant(I,A,Grant_{UI})$. The Authorization Server, after receiving and verifying the forwarded grant, generates an access token and sends it back to the IoT device, which is represented by $AccessToken(A \rightarrow I)$ and $Token_{AI}=GenerateToken(A,I,Grant_{IA})$. This access token serves as a secure credential that the IoT device uses to authenticate itself to the Resource Server.

In the subsequent phase, the IoT device, now armed with the access token, sends this token to the Resource Server, marked as R. This transaction is captured by $AccessToken(I \rightarrow R)$ and $Token_{IR}=SendToken(I,R,Token_{AI})$. The Resource Server, upon receiving the token, verifies its validity and authenticity. Once verified, the Resource Server provides access to the protected resource to the IoT device, as indicated by $ProtectedResource(R \rightarrow I)$ and $Resource_{RI}=AccessResource(R,I,Token_{IR})$. This step ensures that the IoT device can access the necessary data securely. Throughout this process, standard security protocols such as SSL and SSH are assumed to secure the communication channels between all entities, ensuring that data integrity and confidentiality are maintained. This robust

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

mechanism not only enhances security but also ensures that access to sensitive data is controlled and monitored, providing a reliable framework for managing data access in cloud-based environments.

One potential improvement is the incorporation of dynamic access control mechanisms. By leveraging dynamic policies, the system can adjust access rights based on context, user behavior, and real-time analysis. This can be mathematically modeled using a function f that dynamically updates access permissions P based on variables such as time t, location l, and user activity a:

$$P_{new} = f(P_{current}, t, l, a)$$

This ensures that access control is adaptive and responsive to changing conditions. Additionally, we can define the dynamic policy function as:

$$f(P_{current}, t, l, a) = P_{current} (1 + (w_1 t + w_2 l + w_3 a)/(w_1 + w_2 + w_3))$$

where w_1, w_2, and w_3 are weights assigned to time, location, and activity respectively.

Moreover, incorporating audit logs and monitoring tools into the system can significantly improve transparency and accountability. By keeping a detailed log of all access requests, authorizations, and data retrievals, the system can provide a comprehensive audit trail. This can be represented by the equation:

$$Log = \{(t_i, user_i, action_i, resource_i, status_i)\}_{i=1}^{n}$$

where t_i denotes the timestamp, $user_i$ represents the user involved, $action_i$ signifies the action taken (e.g., request, grant, access), $resource_i$ indicates the resource in question, and $status_i$ denotes the outcome of the action.

$$Req = \{(t_i, user_i, resource_i, purpose_i)\}_{i=1}^{m}$$

where $purpose_i$ indicates the purpose of the access. The authorization grant equation can be expressed as:

$$Grant = \{(t_i, user_i, resource_i, validity_i)\}_{i=1}^{k}$$

where $validity_i$ represents the validity period of the grant. For data retrieval actions, the formula can be:

$$Data_{retrieval} = \{(t_i, user_i, resource_i, data_i)\}_{i=1}^{p}$$

where $data_i$ is the retrieved data content. Monitoring these logs for anomalies involves a detection function D defined as:

CAHIERS MAGELLANES-NS
Volume 06 Issue 2
2024

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

$$Anomaly = D(Log) = \{(t_i, user_i, action_i) \mid action_i \mathrel{!=} expected_{action}\}$$

where $expected_{action}$ is the predefined legitimate action for the user. Implementing real-time monitoring and alerting mechanisms can further enhance the system's ability to respond to potential security threats promptly. These enhancements ensure that the proposed mechanism not only maintains high security and privacy standards but also provides robust tools for compliance and risk management.

The proposed solution can be implemented in a modular manner, with each phase being constructed separately and subsequently integrated. This will expedite the development process. Additionally, the resolution of obstacles will facilitate the process of debugging. The solution that has been presented will be elaborated upon in two subsections. The initial subsection will provide an explanation of the functionality of the smart contract and the manner in which it performs authentication during the initial phase. The second subsection delves into the authentication interaction between the user and the IoT device following the successful completion of the initial phase.

3.3 Phase 1: Smart Contract

The initial phase initiates when the user authenticates with the smart contract to establish their legitimacy. The pseudo-code provided below provides a description. The admin user is the sole legitimate user in this iteration of the smart contract. In other words, the admin user is the individual who deployed this smart contract on the blockchain. It is possible to increase the number of users by incorporating an addUser() method into the smart contract.

The Smart Contract of the Proposed Solution:

```
pragma solidity ^0.8.0;
contract Login2 {
   address private owner;
   bytes32 private hash;
   uint256 private random_number;
   event LoginAttempt(address indexed admin, bytes32 hash);

   constructor() {
      owner = msg.sender;
   }

   function login_admin() private {
      require(msg.sender == owner, "Unauthorized: Only the owner can login.");

      random_number = uint256(keccak256(abi.encodePacked(block.timestamp,
block.difficulty, msg.sender))) % 100 + 1;
      hash = keccak256(abi.encodePacked(msg.sender, block.timestamp,
random_number));
```

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

```
      emit LoginAttempt(msg.sender, hash);
   }
}
```

Users utilize their Ethereum client to invoke the login admin method in order to authenticate. The login admin function is protected from public usage and requires no parameters. This is due to the fact that a modifier verifies the sender's Ethereum address, ensuring that only authorized users can access it. When this function is invoked by a verified user, the login admin generates a random hash using the rand function. Then, the login admin generates a token by hashing the user's Ethereum address, block time, and the random hash generated in the most recent step. Subsequently, an event is triggered to transmit the token and the authenticated user's address to the IOT device and user, enabling them to proceed with phase 2.

3.4 Phase 2: User-IOT Interaction

The Ethereum address of the authorized user and an authentication token are received by the user and the IOT device upon the successful completion of phase 1. Phase 2 establishes a connection between the two entities. Please be advised that this solution presupposes that the user is aware of the address of the IOT device, which is either the IP address or the domain name. In the event that this is not the case, the device address can be transmitted via the LoginAttempt event.

User Side Implementation Flow: Initially, the user's script establishes a connection to the Ethereum blockchain by utilizing the web3 Ethereum client and nodejs to monitor events emanating from the deployed smart contract in phase 1. The user's script accesses the keystore directory, which contains the user's secret keys, and extracts the private key when the event has arrived. Please be advised that the script requires the

In order to execute this action, the key must be passed. Subsequently, the script extracts the public key from the private key. The Keccak256 algorithm is employed to hash the public key. The Ethereum address of the user is represented by the final 40 bytes of the resulting hash. This Ethereum address is compared to the one obtained from the smart contract event. This is the official method for obtaining an Ethereum address from a keystore directory. It is important to note that the Ethereum address is used in place of the public key, as it is more convenient and shorter. This is accomplished by utilizing keythereum to access the private key and elliptic to derive the public key. Those Node.js libraries are available for download online [15].

After verifying that the Ethereum address obtained from the smart contract corresponds to the user's address. Constructing the authentication message is initiated by the script. The message can be summarized as follows:

$$message = [token + src_{ip} + Auth_{dur} + Pub_K]$$

where *token*: is the token that was received from the smart contract.
src_{ip}: is the IP address from which the user will establish a connection.
$Auth_{dur}$: The duration of the authentication's validity before it is revoked and a new authentication is

CAHIERS MAGELLANES-NS
Volume 06 Issue 2
2024

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

necessary
The public key of the user's Ethereum account is denoted as Pubk. Ultimately, the private key associated with the user's Ethereum account is employed to sign the message. The authentication package that follows is transmitted to the IoT device.

$$message + Signature + Pub_K$$

IOT Side Implementation Flow: The IOT device script commences in a manner that is comparable to the user script. It establishes a connection with the smart contract that was implemented in phase 1 and monitors for the desired event. The script obtains the Ethereum address and authentication token of the authenticated user when the event occurs. The script awaits the user's authentication package. The verification phase commences upon the package's arrival. The authentication package is discarded in order to reduce the processing power of the IOT device in the event that any of the verification steps are unsuccessful. The current step must be verified before the process advances to the subsequent step.

Steps of the Verification Phase:

- Verify if the authentication package and message are in the correct format.
- Validate the message signature using the public key.
- Check if the public key in the authentication package matches the one in the message.
- Compare the token in the message with the token from the smart contract.
- Ensure the source IP address in the message matches the source IP address of the sender of the authentication package.[23]
- Hash the public key in the message and extract the last 40 bytes to see if the result matches the Ethereum address from the smart contract.

If all these verification steps are successful, the user is authenticated. Otherwise, the authentication package is discarded. This process adds linear execution complexity to the program, adhering to BigO standards, depending on the input.

$$O(kn) = O(n)$$

Where k is a constant.

4. Testing and Evaluation

This section delves into the security and functionality tests that were conducted following the implementation of the prototype of the proposed solution. Furthermore, the cost of establishing communication between the user and the smart contract is included. It is important to note that "setter" functions typically incur a cost for the user, as they necessitate miners to modify the blockchain, whereas "getter" functions do not incur any time or cost.

In order to modify the attributes of a smart contract, a transaction must be executed on the Ethereum blockchain. The fee is determined in GAS and disbursed in Ether. The owner has the option to determine the quantity of gas necessary when the smart contract is implemented. This transaction is mined first due to an increase in gas prices. It is a compromise between cost and priority. The cost of "21K" gas in

CAHIERS MAGELLANES-NS
Volume 06 Issue 2
2024
ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

December 2017 is $0.01. This quantity of gas corresponds to 2 gwei, which is the standard.

Rapidity of transaction execution: This cost is arguably justifiable, as it offers a tamper-proof block of records and a decentralized authentication platform. Alternatively, the alternative solution would be to entrust your data to a centralized entity, which would increase the risk of a single point of failure and the loss of sensitive data. A third alternative is to operate a self-owned decentralized database, which incurs additional expenses for maintenance and administration.

The fluctuating value of Ethereum can present a challenge for smart contract users. Ethereum's value has reached its maximum during a specific period following the deployment test of the smart contract in this paper. Subsequently, it has returned to its standard price. A drawback that impacts the stability of smart contract usage is the fluctuating price of cryptocurrencies. Nevertheless, Ethereum intends to resolve this issue through its forthcoming consensus algorithms.

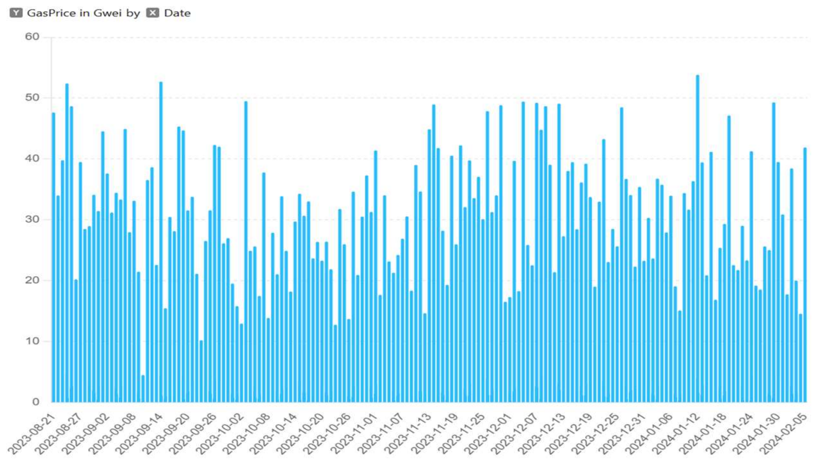

Figure 2: Daily Ethereum GasPrice in Gwei (Aug 21, 2023 - Feb 5, 2024).

The provided bar chart in figure 2 visualizes Ethereum's gas prices in Gwei over the period from August 21, 2023, to February 5, 2024. Each bar represents the daily average gas price, showing significant fluctuations in the cost of executing transactions on the Ethereum network. In the initial weeks, gas prices are observed to be relatively high, frequently exceeding 40 Gwei, indicating substantial network activity and congestion. As we move into October and November, a slight decrease in the average gas prices is noticeable, suggesting a period of reduced demand or improved network efficiency.

However, from mid-December through January, there is a resurgence in gas prices, with several spikes reaching above 50 Gwei, possibly due to increased transaction volume or specific events that caused temporary congestion. This period highlights the volatility and sensitivity of gas prices to network conditions. The chart demonstrates the dynamic nature of Ethereum's network, where gas prices can vary significantly day-to-day based on user demand and network capacity. This visualization is essential for understanding the cost trends associated with Ethereum transactions over time.

CAHIERS MAGELLANES-NS
Volume 06 Issue 2
2024

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

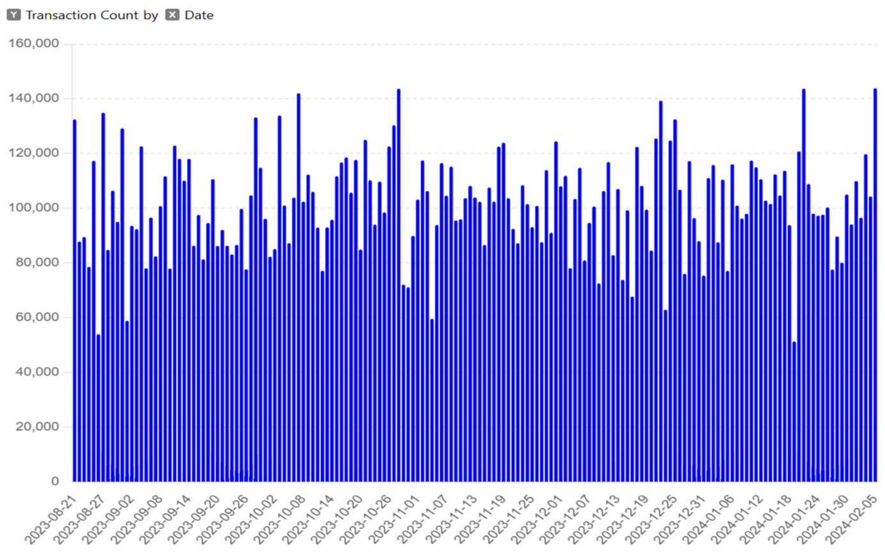

Figure 3: Daily Ethereum Transaction Count (Aug 21, 2023 - Feb 5, 2024)

The bar chart illustrates in figure 3 presents the daily transaction count on the Ethereum network from August 21, 2023, to February 5, 2024. Each bar represents the total number of transactions processed on the network each day. Throughout the period, the transaction count shows a consistent pattern, generally ranging between 60,000 and 140,000 transactions per day. Notable peaks can be observed sporadically, indicating days of higher activity which could be due to specific events, network upgrades, or increased usage of decentralized applications (DApps) and services on Ethereum.

In the initial period of late August and early September, there are frequent spikes in transaction counts, often exceeding 120,000 transactions per day. This suggests periods of high network usage, which could correspond to market movements or significant events in the cryptocurrency space. As we move towards the end of the year, particularly in December and early January, the transaction count becomes more volatile, with several days surpassing 140,000 transactions. This could reflect heightened activity during the holiday season, possibly due to increased trading volumes or other blockchain-related activities.

Overall, the chart highlights the robust and active usage of the Ethereum network over the months, showcasing its ability to handle a substantial number of transactions daily. This level of activity underscores the importance of Ethereum as a leading platform for smart contracts and decentralized applications.

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

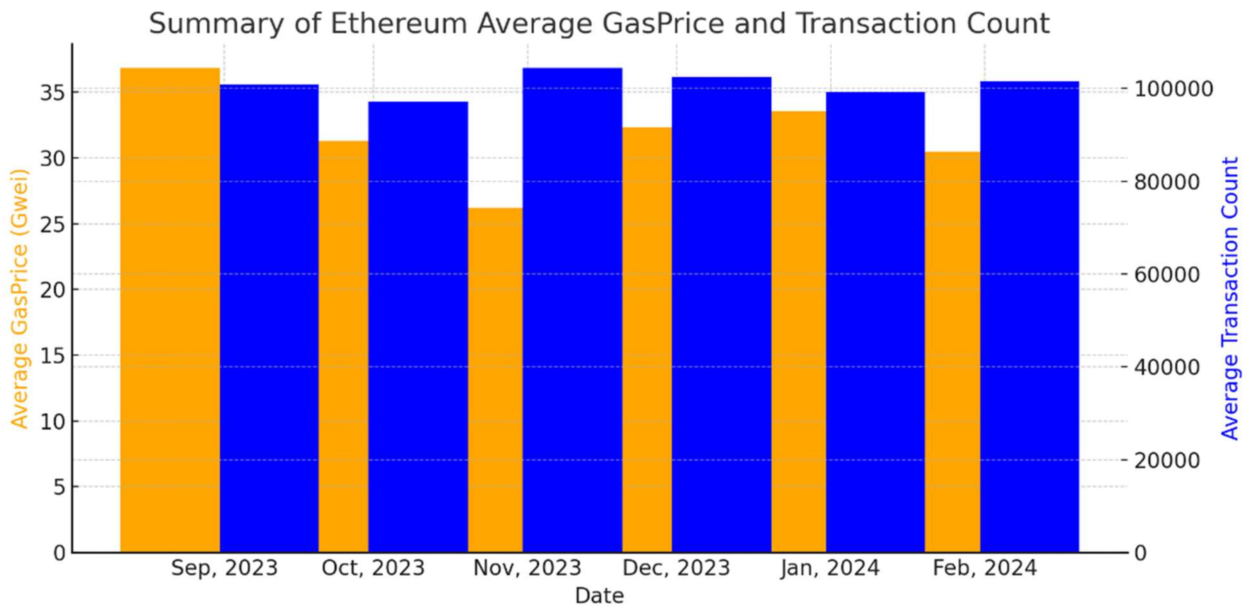

Figure 5: Summary of Ethereum Average GasPrice and Transaction Count (Sep 23 - Feb 24)

The summary bar chart illustrates the monthly average values of Ethereum's gas prices and transaction counts from September 2023 to February 2024. The orange bars represent the average gas prices measured in Gwei, while the blue bars depict the average transaction counts. This dual-axis chart provides a comprehensive overview, showcasing how these two critical metrics fluctuated over the observed period.

In September 2023, the average gas price was slightly higher compared to the subsequent months, indicating increased network activity. Transaction counts remained relatively stable, with a notable increase in January 2024, reflecting a surge in network usage during that month. The chart effectively highlights the relationship between gas prices and transaction volumes, offering valuable insights into Ethereum's network performance and user activity trends over the six-month period.

4.1 Costs

The transaction cost, execution cost, and equivalent price in US dollars for the deployment and utilization of the smart contract are presented in Table 1.

Table 1. Gas cost for running functions

Function	Transaction cost	Execution cost	Total cost	Price in USD
Deployment	275146	170461	445623	$0.2134
login admin	64079	42821	106927	$0.0529

The initial rows display the cost of deploying the smart contract, which is performed only once. Deploying the smart contract is evidently more costly due to its writing to the blockchain. Conversely, the login feature is less costly. The login function can be optimized to reduce costs. The current price is

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

due to the fact that it generates the authentication token by hashing, generating a random number, and retrieving the block hash. These prices are indicative of the expense associated with employing the conventional proof-of-work consensus protocol. The costs will decrease significantly as a result of Ethereum's transition to standardizing the proof-of-authority protocol, as the workload of the miners will decrease as well.

4.2 Testing

The testing phase was partitioned into distinct subsections. Initially, the proposed solution undergoes manual testing to guarantee its security, performance, and robustness. In addition to the ideal test cases, manual tests incorporate malicious scenarios. Secondly, static analysis tools are employed to conduct an automated security assessment of the smart contracts.

Manual Testing: The ideal scenario was tested first after the solution prototype was run. The smart contract function is invoked by a legitimate user who logs in as the administrator using their MIST Ethereum client. The smart contract simultaneously transmits the user's Ethereum address and the authentication token to the user and IoT device. The initial phase was completed on a private blockchain in less than four seconds, as indicated by the test. The user subsequently establishes a connection with the IoT device by transmitting the authentication package delineated.

- The IOT authentication script's verification steps were bypassed through the execution of a few malicious attacks, as detailed below:
- The replay attack was unsuccessful due to the fact that the source IP of the attacker must be identical to the source IP of the signed authentication message.
- The script failed to modify the signed authentication message because it verifies the message signature.
- The attacker's authentication package was unsuccessfully injected, as the public key should correspond to the Ethereum address of the legitimate user.

It is possible for a man-in-the-middle to intercept outgoing authentication packages. Nevertheless, integrity is safeguarded by the fact that the signed authorization message cannot be altered. The test phase's results demonstrate that this solution is secure, provided that the user's keypairs are not compromised. Upon successful completion of the authentication process, the authentication tokens should be invalidated and replaced. The subsequent actions are beyond the scope of this paper.[22]

The test environment is subject to change when the current solution is being tested. Initially, the majority of tests are conducted on a private Ethereum blockchain. This facilitates the mining and validation of transactions. Afterward, it is advised to employ Rinkeby instead of Ropsten when testing the smart contract on the public Ethereum blockchain test-net, as it employs Proof-of-Authority rather than Proof-of-Work, which is the method employed by Ropsten. In addition, this will facilitate the public testing approach.

CAHIERS MAGELLANES-NS
Volume 06 Issue 2
2024

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

The current method employed in the Ethereum main network and the Ropsten network to confirm transactions is the proof of work concept. A mathematical puzzle is solved by a miner in order to validate the transaction and receive an incentive. The execution of this method necessitates a significant amount of processing power. In contrast, the Rinkeby test-net's proof of authority is contingent upon a set of explicitly authorized nodes, rather than miners resolving mathematical problems. Consequently, it is regarded as the future of mining techniques, as it does not necessitate as much processing power [16].

Automated Testing: Static Security Analysis: ConsenSys conducted a security assessment through static analysis using mythril. It employs concolic analysis to identify security vulnerabilities in smart contracts. It is capable of functioning in both whitebox and blackbox testing scenarios. A whitebox testing was conducted on the smart contract source code due to its availability. Mythril did not encounter any complications. Furthermore, a control flow graph is generated for the smart contract source code to guarantee that all potential paths are examined. The Ethereum Laser Symbolic Virtual Machine is employed to generate the graph.

Lastly, performing formal method tests on the smart contract to ensure that all potential execution paths are anticipated and covered is another metric that can be suggested for future testing. The current endeavors are documented in [17].

4.3 Evaluation

To assure the quality of the proposed solution, the next step is to compare it to previous solutions. The evaluation metric is based on whether the offered authentication scheme solved problems occurring in the other authentication mechanisms proposed for the IOT devices.

Table 2. Comparing and evaluating authentication solutions

	Auth/device	Oauth2	Auth0	Blockstack	Paper sol.
Availability	C	X	X	X	C
Scalability	X	X	C	C	C
Decentralization	C	X	X	C	C
Tamper proof	X	X	X	C	C

The proposed solution is compared in Table 2 based on availability, scalability, decentralization, and tamper-proof. The evaluation metrics are defined with greater precision for this comparison. Availability is defined as the absence of a single point of failure and the removal of the bottleneck. The term "scalability" is employed to delineate the additional overhead that the application incurs when additional devices are incorporated. Decentralization is the capacity of the authentication application to function independently of a central entity that could potentially disrupt the system in the event of its failure. Tamper proof is the guarantee that saved data and transactions cannot be tampered with once they have been recorded in the system's logs.

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

5. Conclusion

In this paper, we have suggested a blockchain-based solution that enables users to securely access IoT devices by proving their authentication. We illustrated how our methodology surpasses the deficiencies of existing authentication protocols. We demonstrated that our blockchain-based solutions, which utilize Ethereum smart contracts, can enhance current methods by enabling decentralization and tamper-proof records. Using available IoT devices and technologies, we developed and executed our solution in accordance with real-world scenarios. In particular, we demonstrated the process of successfully authenticating legitimate users in order to access their IoT devices. Additionally, we demonstrated that our methodology could withstand crafted attacks that were designed to hijack legitimate sessions and brute force credentials. In the future, we intend to expand the proposed approach to encompass a vast number of IoT devices and end users by implementing a massive scale access and authentication system. Additionally, we intend to evaluate the application's performance in terms of scalability and cost (or gas) consumption on a genuine Ethereum blockchain network. We are also taking into account the monetization aspects of IoT devices and their data, which involve the payment of usage through a crypto-token of ether.

6. References

1. Al-Bassam, M.: SCPKI: a smart contract-based PKI and Identity System (2017).
2. Amadeo, R.: Don't trust OAuth: why the "Google Docs" worm was so convinc-ing. https://arstechnica.com/security/2017/05/dont-trust-oauth-why-the-google- docs-worm-was-so-convincing/
3. Evans, D.: The Internet of Things. How the Next Evolution of the Internet Is Changing Everything (2011)
4. George V.: A Next-Generation Smart Contract and Decentralized Application Platform (2018)
5. Hirai, Y.: Formal Verification of Ethereum Contracts (2018)
6. Liu, J., Xiao, Y., Chen, C.: Authentication and access control in the Internet of Things. In: 2012 32nd International Conference on Distributed Computing Systems Workshops, pp. 588–592 (2012)
7. Ali, M., Shea, R., Nelson, J.: Blockstack: A New Internet for Decentralized Appli-cations (2017)
8. McKinney, J.: Light client protocol (2017)
9. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
10. Peyrott, S.: An Introduction to Ethereum and Smart Contracts: an Authentica-tion Solution. https://auth0.com/blog/an-introduction-to-ethereum-and-smart-contracts-part-3/
11. Symantec: Latest Intelligence for June 2017. https://www.symantec.com/connect/blogs/latest-intelligence-june-2017

ISSN:1624-1940
DOI 10.6084/m9.figshare.260185
http://magellanes.com/

12. Minerva, R., Biru, A., Rotondi, D.: Towards a definition of the Internet of Things (IoT) (2015)
13. Gusmeroli, S., Piccione, S., Rotondi, D.: IoT access control issues: a capability based approach. In: Sixth International Conference on Innovative Mobile and Inter- net Services in Ubiquitous Computing (2012)
14. Sandoval, K.: Why OAuth 2.0 Is Vital to IoT Security. https://nordicapis.com/why-oauth-2-0-is-vital-to-iot-security/
15. Swamy, N., Hriţcu, C., Keller, C., Rastogi, A., Delignat-Lavaud, A., Forest, S., Bhargavan, K., Fournet, C., Strub, P., Kohlweiss, M., Zinzindohoue, J., Zanella-B́eguelin, S.: Dependent types and multi-monadic effects in F*. SIGPLAN Not. **51**, 256–270 (2016)
16. Szabo, N.: Smart Contracts: Building Blocks for Digital Markets (1996)
17. The ethereum: Accounts, Addresses, Public and Private Keys, and Tokens. https://theethereum.wiki/w/index.php/Accounts, Addresses, Public And Private Keys, And Tokens
18. Thomson, D.: IoT and the problem of identity. https://www.symantec.com/connect/blogs/iot-and-problem-identity
19. Tosh, D., Shetty, S., Liang, X., Kamhoua, C., Njilla, L.: Consensus protocols for blockchain-based data provenance: Challenges and opportunities. In: 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Con- ference (UEMCON), pp. 469–474. IEEE (2017)
20. Mrs. Misbah Kousar, Dr. Sanjay Kumar, Dr. Mohammed Abdul Bari," A Study On Various Authentication Schemes In Iot To Provide Security", Educational Administration: Theory and Practic,ISSN No : 2148-2403 Vol 30- Issue -6 June 2024
21. Farhan Ali Baig, Hajera Zia, Dr. Mohammed Abdul Bari, "Decentralised Social Media Platform Using Blockchain Technology", International Journal Of Research In Electronics And Computer Engineering (IJRECE), Vol. 10 Issue 2 April-June 2022; ISSN: 2393-9028
22. Mohammed Azharuddin, Aslam Khan, Syed Faizan Ali, Dr. Mohammed Abdul Bari, "Web 3.0 application using blockchain", International Journal of Research In Electronics And Computer Engineering (IJRECE), Vol. 10 Issue 2 April-June 2022; ISSN: 2393-9028
23. Anas Bin Yahiya Qureshi, Safi Ur Rahman Pasha, Mohammed Sameer, Dr. Mohammed Abdul Bari, "Collaborations with Smart contracts using Blockchain", International Journal Of Research In Electronics And Computer Engineering (IJRECE), Vol. 10 Issue 2 April-June 2022; ISSN: 2393-9028