A Decentralized Mechanism to Decouple Vendor-Specific Access Management from IoT Devices Using Blockchain Technology, Smart Contract, and Wallets

Jacobs Jacob Chakola[1](✉), Garima Sinha[2], and Deepak K. Sinha[2]

[1] Cyber Security, CSE, Jain (Deemed-to-be) University, Bangalore, India
Jacobschakola@gmail.com
[2] CSE, Jain (Deemed-to-be) University, Bangalore, India

Abstract. This paper provides a mechanism for distributed access management and authorization transfer for communicating between devices belonging to different wallets and the same wallet. This paper also discusses segregating the IoT network into different groups or levels for fine-grain control. This approach allows Access control and intersystem network communication between authenticated devices and authenticated devices residing in the same system, using the devices layer, smart contract, and wallets. The Device layer is used to transmit data from one device to another, regardless of the wallet they belong to. The smart contract acts as the distributed command and control for access management. The wallet stands as the owner of the IoT in the system. The wallets are generally owned by individuals or end consumers of the IoT product. The approach can be split into three phases: initialization, device authentication, and device-to-device communication.

Keywords: smart contract · blockchain technology · IoT · access management · access control · decentralized mechanism

Abbreviation

A_{PK}	*Public key of device A*
A_{IK}	*Private key of device A*
W_{PK}	*Public key of the wallet that owns the IoT device*
W_{IK}	*Private key of the wallet that owns the IoT device*
SC_{AD}	*Address of the smart contract deployed by IoT device owner*
RG_C	*Registration certificate*
IZ_C	*Initialization certificate*
$Auth_C$	*Authentication Certificate*
Mac_A	*Mac address of IoT devices A*
Mac_B	*Mac address of IoT devices B*
IP_A	*IP address of IoT devices A*
IP_B	*IP address of IoT devices B*

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023 2023
O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14104, pp. 538–554, 2023.
https://doi.org/10.1007/978-3-031-37105-9_36

PL_A	*Privilege-level of IoT devices A*
$AuthPacket_A$	*Auth Packet of device A*
$AuthPacket_B$	*Auth Packet of device B*
AT_C	*Access transfer certificate*
AG_C	*Access transfer certificate*
PLT	*level of privilege the transferred user has on the IoT device*
Wt	*The Wallet that is being granted access to the IoT devices.*
WtSC	*The smart contract address owned by Wt.*
Wt_{PK}	*Public key of the Wallet the access is being granted*
Wt_{IK}	*Private key of the Wallet the access is being granted*
WtAddress	*the address of the wallet that receives the NFT*
NFT	*Non-Fungible Token Standard*
NFT ID	*NFT token ID*

1 Introduction

The IoT paradigm represents interconnected networks and heterogeneous devices. The central element of IoT is to expedite sharing of the constrained devices' resources and information with other entities. With the advent of 5G technology, IoT has gained a rapid endorsement in many domains. Where devices with constrained resources are interconnected either through private or public networks, ranging from embedded devices containing sensors such as smart home appliances to vehicle-to-vehicle in supply chain scenarios. The devices or "things" in IoT can be accessed remotely to perform the specified functionality. The solutions for access management are more analogous to the server-client architecture usually owned and managed by the vendors of the IoT devices, designed to manage IoT in the same ecosystem in a centralized fashion.

However, the solutions for IoT access management confront additional challenges as the "things" or devices are conventionally constrained when it comes to memory and processing. Inevitably, the access management solutions must acclimate constrained architecture.

Different vendors enforce their ecosystem and access control that can be used only with their products. That Leads to limited or no interoperability pertaining to access management. This reliance implies the assumption that the vendor's services are always available. This paper intends to decouple the IoT devices and vendor-specific access management using smart-contract deployed on the blockchain, which stands as the command and control for access management. The distributed nature of blockchain technology addresses the constant availability of authentication and access management services regardless of the device vendor.

1.1 Paper Structure

The paper is systematized as follows: Sect. 2 provides a short primer on the workings of blockchain technology, such as hash Pointer, the different consensus algorithms, Smart Contracts, NFTs, and as well as the Limitations of blockchain. Section 3 discusses and

examines some related works. Section 4 explains the distributed access management and transfer of authorization mechanism by splitting the approach into four use cases. Section 6 discusses the value proposition of the mechanism. Section 7 discusses the Security Analysis. Finally, Sect. 8 concludes the paper.

2 Background

2.1 Blockchain

Ever since Satoshi Nakamoto made blockchain technologies public debut in 2008 with the whitepaper Bitcoin [1], blockchain has skyrocketed in popularity, expanding beyond the cryptocurrency paradigm into decentralized applications. For most people, blockchain, and bitcoin are synonymous even to this day; blockchain is a distributed, decentralized, immutable ledger that records assets and transactions in a peer-to-peer network with a timestamped block of data chained together as a link list of hash pointers [2].

Blocks in the blockchain act as data storage for the blockchain, each block in the blockchain stores all the transaction information as a database. The critical property that fundamentally distinguishes it from conventional databases is its public verifiability. Public verifiability is achievable by virtue of integrity and transparency. Blockchain facilitates transparency by permitting anyone to verify the state of data and how it was updated; as for integrity, it means data received is not corrupt.

It accomplishes data integrity and authenticity by delivering robust cryptographic proof using elliptic curve cryptographic (ECC) and SHA-256 hashing.

A block in the blockchain contains the hash of the previous block in the header. The genesis block is the first block in the chain and is usually hardcoded (Fig. 1).

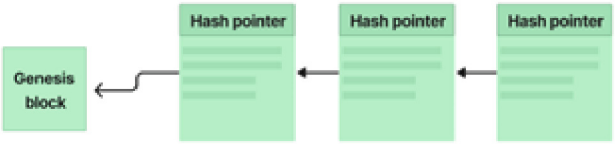


Fig. 1. Simplified overview of a blockchain

The blockchain is a trustless architecture where no third parties or centralized authority could corrupt the data in the blockchain, and since it is distributed, there is no single point of compromise.

The different nodes communicate using the gossip protocol. Which inherently creates propagation delays in the network. Potentially create blocks simultaneously.

Consequently, creating two separate branches of the blockchain (Fig. 2) leads to some miners mining on top of the forked branch, fundamentally going against the nature of the blockchain. However, these forked branches will be discarded as the branch with the most combined difficulty will be selected, not the branch with more blocks. The discarded or invalid chains are usually called orphan blocks.

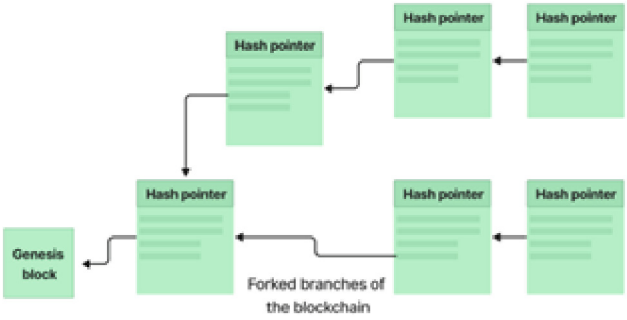


Fig. 2. Forked branch of the blockchain

2.2 Smart Contracts

Blockchain like Ethereum has an Ethereum virtual machine, which makes it a Turing Complete blockchain. This allows Ethereum blockchain the capability to understand and implement any future agreement in the form of smart contracts. Smart contracts are transaction protocols or rather a self-operating computer program or self-executing code when a predetermined condition is satisfied and resides on the blockchain. Which eradicates any involvement of a trusted intermediary [3]. This paper ERC-721: (Non-Fungible Token Standard) for NFT minting [4].

2.3 Hash Pointers

Hash pointer serves two purposes. It uniquely identifies the previous block of data. It also confirms that the data has not been tampered with; due to the properties of the hashing functions. It acts as a tamper-evident log by comparing the previous block's hash to the hash in the next block's header, proving that the block has not been tampered with (Fig. 1).

2.4 NFT

NFTs are non-fungible tokens on the blockchain that are distinguishable with a unique identification code. The NFTs have meta data that is part of the token. These NFTs, unlike cryptocurrencies, are not identical and cannot be exchanged at equivalency [5].

2.5 Consensus

Consensus is the mechanism that achieves the necessary agreement on the blockchain network's state. In the case of bitcoin, the mechanism is proof of work (PoW) [6], also called mining, where the miner computes a hash with leading zeros to meet the difficulty target; once the block is validated, it gets written onto the blockchain and becomes immutable.

There are numerous algorithms for consensus.

Proof-of-Stake (PoS) verifies blocks by staking coins as collateral to become "validators".

Validators are randomly selected to validate the block [7].

Proof of capacity (PoC) verifies blocks by storing possible solutions on the miner's hard drive. The more substantial the drive, the more possible solutions stored. In turn, the greater the chance of matching the required hash. This approach is more analogous to winning a lottery [8].

Proof-of-Activity (PoA) verifies blocks using the two consensus algorithms, PoW and PoS. The miners authenticate a transaction depending on the number of coins staked, using the PoW consensus. The PoA algorithm is designed to reduce the overall power consumption [9].

Proof of burn (POB) operates on the principle of "burning" or sending coins to a verifiably inaccessible wallet address, which burns the coins as the coins sent to this wallet cannot be spent. Doing so grants the privilege to write to block proportionally to the burnt coins [10].

Proof of History (PoH) [11] and Proof of Elapsed Time (PoET) [12] are similar, encoding time passage to determine consensus. PoET is conventionally used in permissioned blockchains, requiring prospective participants to identify them before joining.

Each node in the blockchain sleeps for an assigned random wait time.

A node with the shortest assigned wait time wakes and writes a block to the blockchain.

PoW and PoS consensus algorithm is designed to prevent a situation where a group of miner's controls more than 50% of the blockchain's mining hash rate, including halting new transactions and changing historical blocks granting them the ability to double-spend.

2.6 Limitations of Blockchain

Although blockchain solves some problems in IoT, it is far from a perfect solution. Blockchain still has some limitations. The capabilities and effectiveness of blockchain can be conquered with further investigation.

The Low transaction throughput and high latency resulted in inadequate performance due to the poor scalability of the consensus mechanisms.

The high-power consumption of the blockchain as it scales on constrained devices is due to the complex mining algorithms. However, power-efficient blockchain algorithms could alleviate such difficulty.

Since all transactions on a public/permissionless blockchain are public and participating devices are identified by their public key, they are susceptible to pattern analyses and establishing a connection between addresses, inevitably inferring information and identity of the parties involved [13].

3 Related Work

In [14] proposed a scheme to authenticate, which requires the storage of Symmetric key cryptography on a cloud server. The manufacturers of wearable and smartphones save the devices' details and sensory data on the cloud. The problem with this approach is that all the devices' details, like the symmetric key and the sensory data, are susceptible to a single point of failure attack.

In [15], the proposed approach uses RSS signal variation and RSS-trace concept to prevent physical security attacks like impersonation and device capturing for proximity-based authentication between smartphones and IoT devices. The limitation of this mechanism is that it requires proximity to the IoT and that the authentication data resides on a central local server. As a consequence, it creates a single point of failure.

In [16], the proposed authentication mechanism uses PUFs (physical unclonable Functions) without storing secret keys on centralized servers, which would make it impossible to impersonate but requires the PUF-based authentication data to reside on the server nodes. However, it would make the approach susceptible to single-point failures.

In [17], the proposed two-factor authentication mechanism uses physical properties to identify the IoT devices to prevent impersonation attacks and side-channel attacks. The main drawback to the approach is storing authentication data on a centralized server.

In [18], proposed authentication and a secure communication mechanism between industrial IoT devices. The approach utilizes hash functions and XOR operations for transmitting sensory data and authentication. The limitation of this mechanism is that it mandates the storage of authentication data on a local server, creating a single point of failure.

The mechanism proposed in [19] has three main components, the IoT devices, group leader, and subscriber server. That allows authentication of groups of devices before entering into the system. The registration and authentication data is on the subscriber server. The limitation is that the authentication data of various groups reside on the subscriber server, creating a single point of failure.

[20] proposes a decentralized authentication mechanism using blockchain technology to circumvent several security issues in IoT. However, the primary constraint is that communication is limited to devices within the same system, making it impractical in the IoT paradigm as it is vital for many distributed applications of IoT that require communication between devices of different systems.

4 Proposed Mechanisms

The primary purpose of this work is to provide distributed access control and management for IoT devices irrespective of the vendor allowing for interoperability. This paper provides a mechanism to authenticate IoT devices with IoT devices belonging to the same or different systems. This approach discusses a secure way to transfer access to another user who might not be part of the IoT system as well as segregation of the IoT network.

4.1 Segregation of the IoT Network

The segregation of the IoT network into different levels allows greater control over which devices can communicate with other devices. Only devices that are allowed can intercommunicate between levels. So, if a device or network level gets compromised, it prevents the entire system from being compromised. This segregation should prevent vertical escalation. The network-level segregation acts as the privilege level the IoT device has on the network (Fig. 3).

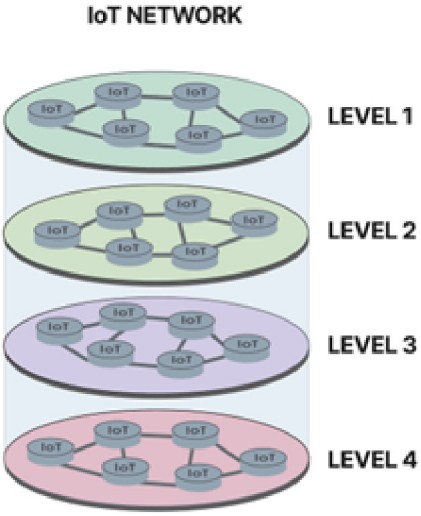


Fig. 3. Segmentation of the IoT network

The network level and its privileges are determined by the Wallet owner. The network level refers to the which segment of the network the IoT reside in, as this determines which other IoT it can intercommunicate. Together with the privilege the IoT has creates the privilege-level (PL_A). An example of what the IoT could have privilege is reading and writing data to a shared database or which IoT devices could communicate with an IoT even on the same network-level. As some IoT device could be programed only communicate with certain other IoT devices. So, there can further segregation within the same network-level. PLT refers to the level of privilege the transferred user has on the IoT device or if the IoT could communicate with other IoT devices own by this wallet/user. P_{code} is 16 characters passcode this was is used to prevent replay and impersonation attacks, each P_{code} is tied to a PLT and privilege-level (PL_A). Not only does this prevent vertical escalation, it allows the IoT to be deployed in environment that require complex segregation of the IoT network for security reasons.

4.2 System Architecture

The mechanism has four components: the device layer, A relay node, Smart Contract, and wallets. The device layer is principally to communicate between the authenticated

devices to establish secure communication. The relay node subscribed to the events emitted by the smart contract and relays it to the IoT, there can be more than one reply nodes deployed to provide redundancy as it only purpose is to send the authentication certificate ($Auth_C$) which is emitted by the smart Contract to the IoT devices, this allows for less complexity on the IoT devices as they do not have to directly subscribe to the smart contract, The relay node resides inside the IoT network and is not deployed on the cloud. The Smart Contract is the entity that provides the authentication certificate ($Auth_C$); the authentication between the devices uses this issued authentication certificate ($Auth_C$). The smart Contract acts as distributed access management for the registered IoT devices. The wallet acts as the owner of the IoT device, and the user interacts with the IoT using the wallet. The approach has three main phases: registration, device authentication, and device-to-device.

Registration phase: The essential principle of this phase is to establish ownership between the IoT (device) and the user (wallet). The user obtains the IoT device's public key (A_{PK}) and mac address and sends a registration certificate (RG_C) containing the public key of the user's wallet (W_{PK}) this public key is not the same as the ethereum public key which is used for making transactions and the smart-contract address (SC_{AD}) as well as a 16 characters passcode (P_{code}) which is tied to a network-level. This registration certificate is signed using the public key of the IoT device. The registration phase mandates the IoT to receive a signed certificate with its public key to establish ownership. The IoT only accepts registration certificate that is signed by its public key as it stores the certificate to use in further validation. The IoT accepts registration certificate only once and would require reset for the devices to accept again. Note that last 40 bytes of the keccak-256 hash public key is the Ethereum address

$$RG_C = A_{PK}\{W_{PK} + SC_{AD} + P_{code}\}. \tag{1}$$

Device authentication phase: This phase is paramount to the mechanism, as the IoT device receives the authentication certificate ($Auth_C$). The IoT device uses this certificate to authenticate itself with other IoT devices. The wallet sends the initialization certificate (IZ_C) to the smart contract. The initialization certificate (IZ_C) contains the mac address (Mac_A), the public key (A_{PK}), and the IP address (IP_A) of the IoT devices, privilege-level (PL_A). Privilege-level contains the network-level and privilege of the device. The initialization certificate (IZ_C) is encapsulated using the wallet's private key (W_{IK}).The initialization certificate also contains a copy of the IP address (IP_A) of the IoT devices that is not encapsulated using the using the wallet's private key (W_{IK}).The smart contract can be programmed to accept transactions only from the wallet that deployed it on the blockchain or from a few assigned wallets, but this paper will focus on the first option. The smart contract crafts the authentication certificate ($Auth_C$) and trigger an event which contains the authentication certificate ($Auth_C$) as well as the IP address of the device (IP_A). The relay node relays the $Auth_C$ to the corresponding IP address in the emitted event. Once the IoT device receives this, the authentication certificate ($Auth_C$) is validated using the registration certificate.

$$IZ_C = W_{IK}\{Mac_A + A_{PK} + IP_A + PL_A\} + IP_A. \tag{2}$$

$$Auth_C = W_{IK}\{Mac_A + A_{PK} + IP_A + PL_A\}. \tag{3}$$

IoT validating the Authentication Certificate.

using Registration Certificate.

$\Rightarrow$ *The IoT device A applies* W_{PK} *from.*

Registration Certificate on the.

Authentication Certificate.

$$\Rightarrow W_{PK}\{W_{IK}\{Mac_A + A_{PK} + IP_A + PL_A\}\}. \tag{4}$$

$\Rightarrow$ *The IoT devices A check if the Mac Address,*

A_{PK} *and IP Address are the same as its own.*

Device-to-device phase This phase pertains to authenticating the IoT devices with other IoT devices. The device creates an auth packet containing the authentication certificate and the device's IP and mac address. The device's IP and mac address in the pack are signed using the device's private key. The device that receives the broadcast can validate the authenticity by applying the public key of the wallet on the authentication certificate.

The device's public key is provided in the registration certificate, once it obtains the public key of the broadcasting IoT from the authentication certificate. The receiving IoT can validate that the device's IP and mac address provided in both the authentication certificate and the one signed using the device's private key are the same. If all the checks are valid, the IoT device sends back its auth packet signed using the public key of the broadcasting IoT devices. Now since both the IoT devices have each other public keys, they have the facilities to initiate secure communication with each other. The IoT that receives the broadcasting sents a random generated string of characters along with the authentication certificate, which the broadcasting IoT will hash with the passcode (P_{code}) and send back.

Which the receiving IoT verifies by hashing the passcode (P_{code}) with the random string of characters that was send, if they match, this ensure the IoT is not imitated by an adversary on the network,

$$AuthPacket_A = Auth_C \ of \ Device \ A + A_{IK}\{IP_A + Mac_A\}. \tag{5}$$

The receiving Device B applies the W_{PK} *to the.*

authentication Certificate of Device.

$$\Rightarrow W_{PK}\{W_{IK}\{Mac_A + A_{PK} + IP_A + PL_A\}\}. \tag{6}$$

$$\textit{The Device B now applies the } A_{PK} \textit{ it just obtained.}$$

$$\textit{for the Authentication Certificate.}$$

$$\Rightarrow A_{PK}\{A_{IK}\{IP_A + Mac_A\}\}. \tag{7}$$

$$\Rightarrow \textit{The Device B validates if the Mac and.}$$

$$\textit{IP address are the same as the one.}$$

$$\textit{in the Authentication Certificate.}$$

$$\Rightarrow \textit{let } \delta = \textit{random generated string.} \tag{8}$$

$$AuthPacket_B = Auth_C \textit{ of Device } B + B_{IK}\{IP_B + Mac_B + \delta\}. \tag{9}$$

$$\Rightarrow \textit{The Device A hashes } \delta \textit{ with } P_{code}.$$

$$A_{IK}\{B_{PK}\{\textit{hashed value of } \delta \textit{ and } P_{code}\}\}. \tag{10}$$

$$\textit{The Device B validates the hash received from Device A.}$$

$$\textit{By hashing } \delta \textit{ with } P_{code} \textit{ and checking is it match.}$$

Case 1: device-to-device communication between devices in the same system owned by the same wallet.

1. The wallet obtained the public key and Mac address of the IoT device, which is to be registered under said wallet and deployed onto the network.
2. The wallet sends a Registration Certificate Eq. (1) to the IoT.
3. Then the wallet sends an Initialization Certificate Eq. (2) to the Smart Contract.
4. The Smart Contract triggers an event and the relay node relays the Authentication Certificate Eq. (3) to the IoT device.
5. The IoT device upon receiving the Authentication Certificate creates Auth packet Eq. (5) and broadcasts it on the network
6. Upon receiving the broadcast, IoT devices validate the Auth Pack and send back their Auth Packet Eq. (8) along with a random generated string of characters to be hashed with passcode (P_{code}). Once this hash is verified, secure connections between the IoT devices are made (Fig. 4).

In a scenario where the IoT system mandates higher security and finer control or the possibility of the authenticated IoT devices being physically accessed by an untrusted

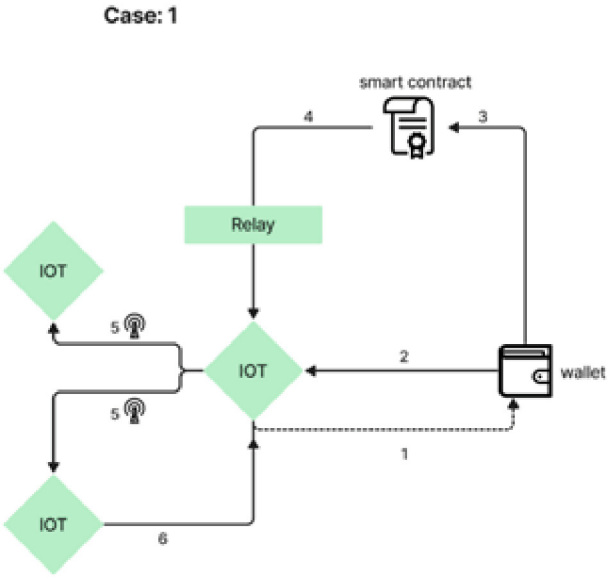


Fig. 4. Device-to-device communication between devices in the same system owned by the same wallet.

Party and the certificate can be stolen. The following steps would mitigate these issues. To increase security in certain networks the IoT device that receives the Broadcasted Eq. (5) can sent the Eq. (8) signed with W_{PK} to the wallet and once the wallet receives the packet, The wallet would apply W_{IK} to extract the Auth packet of device B and then sign it with W_{IP} and send it to the IoT device A.

Case 2: Wallet-to-device communication between wallet and device owned by the same wallet.

Once the device receives the Authentication Certificate and secure communication is established. The Wallet can send packets that are signed with the wallet's private key and then encapsulated with the public key of the IoT device.

1. The packet that was sent from the wallet.

$$\rightarrow A_{PK}\{W_{IK}\{DATA\}\}. \tag{11}$$

2. The IoT device A validates the packet sent from the wallet by applying the device's private key and the public key of the wallet from the registration certificate.

$$\rightarrow A_{IK}\{A_{PK}\{W_{IK}\{DATA\}\}\}. \tag{12}$$

$$\Rightarrow W_{PK}\{W_{IK}\{DATA\}\}. \tag{13}$$

The IoT device has the same facility to send data back in the same manner. This allows the device and wallet to set up a secure connection.

Case 3: device access and communication between wallet and device not owned by the same wallet using non-fungible tokens (NFT).

To allow the IoT device to be accessed by another wallet(user) the steps 1 to 4 from case: 1 must be completed first before the following steps

1. The wallet sends an Access transfer certificate (AT_C) to the smart contract that is encapsulated using the private key of the owner's Wallet except for the wallet address the NFT is sent to which is the same as the wallet the access is being granted (*Wt address*). The Access transfer Certificate also contains public key of the IoT device A (A_{PK}), Mac address of the device A (Mac_A), IP address of the device A (IP_A), public key of the Wallet the access is being granted (Wt_{PK}) this public key is not the same as the ethereum public key which is used for making transactions, the passcode (P_{code}) of the IoT encapsulated using the public key of the Wallet the access is being granted (Wt_{PK}),the passcode (P_{code}) of the IoT encapsulated using the public key of the IoT device A (A_{PK}), level of privilege the user has on the IoT device (PLT). The PLT determines whether the IoT device can be connected to the IoT devices owned by the newly granted wallet(user).

$$\Rightarrow let\ Td = time\ duration\ Wt\ has\ access. \tag{14}$$

$$\Rightarrow let\ \alpha = A_{PK} + Mac_A + IP_A + Wt_{PK} + PLT + Td. \tag{15}$$

$$\Rightarrow let\ \beta = Wt_{PK}\{P_{code} + A_{PK} + Mac_A + IP_A + PLT + Td\}. \tag{16}$$

$$AT_C = W_{IK}\{\alpha + A_{PK}\{P_{code}\}\} + \beta + (Wt\ address) + IP_A. \tag{17}$$

2. The smart contract crafts the Access grant Certificate (AG_C) from Access transfer certificate. Then the smart contract triggers an event that contains an Access grant Certificate (AG_C) and the relay node relays it to the IoT device. The Access grant Certificate (AG_C) contains the data from the Access transfer Certificate and the wallet address the NFT (Wt address) is sent to as well as the NFT token ID (NFT ID). Once the IoT receives the Access grant Certificate it stores it for further use

$$From\ the\ AT_C.$$

$$\gamma = W_{IK}\{\alpha + A_{PK}\{P_{code}\}\} + (Wt\ address). \tag{18}$$

$$\rightarrow AG_C = \gamma + NFT\ ID + IP_A. \tag{19}$$

3. The smart Contract mints a NFT token and sends it to the Wt. The metadata in the NFT contains the time duration the Wt has access to the IoT devices.

$$From\ the\ AT_C\ the\ \beta\ is\ extracted.$$

$$Wt_{PK}\{P_{code} + A_{PK} + Mac_A + IP_A + PLT + time\ duration\}. \tag{20}$$

4. The Wt crafts Access packet and sends it to the IoT devices. The Access packet contains.

$$Wt_{IK}\{A_{PK}\{NFTID + DATA\}\}. \tag{21}$$

$$DATA = random\ generated\ string\ of\ characters. \tag{22}$$

The IoT validates the Access packet by check applying Wt_{PK} provided in the Access grant certificate and checking the NFT metadata using NFT token ID to validate that access time duration has not expired. Since the NFT token ID can be used to track the NFT and who owns it. Then the random generated string of characters in the Access packet is hashed with P_{code} and is sent back to the wallet. The wallet verifies by hashing the generated string of characters with the P_{code} received in the NFT metadata. By doing so wallet can be ensured that the IoT is not imitated by an adversary and secure connection can be established (Fig. 5).

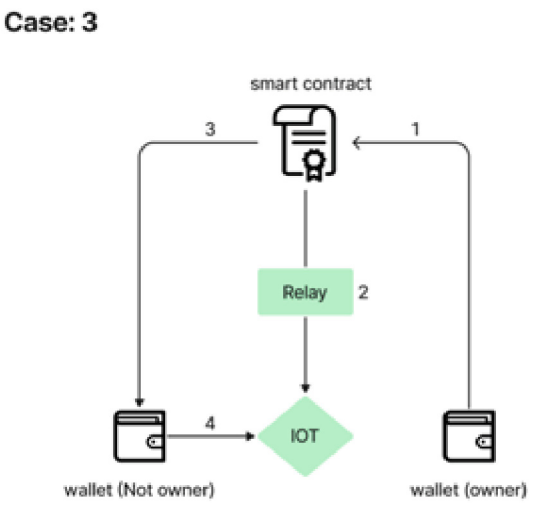


Fig. 5. Device access and communication between wallet and device not owned by the same wallet using non-fungible tokens (NFT).

Case 4: device-to-device communication between different systems belonging to different wallets.

To allow the IoT device to communicate between devices owned by another wallet(user) the steps from case: 3 must be completed first before the following steps. Once the Wt is authenticated to access and the PLT given to the wallet allows the IoT device to connect to the devices owned Wt.

The Wt creates a certificate WtRegistration Certificate which contains the Address of the smart contract (WtSC) owned by Wt as well as a 16 characters passcode (P_{code}) which is tied to a network-level of Wt's network and sends it to the IoT similar to the Registration phase.

And then Wt sends a initialization certificate contains the mac address (Mac_A), the public key (A_{PK}), and the IP address (IP_A) of the IoT devices and the IP address of the device it should communicate with on the Wt system, privilege-level it has on the system owned by Wt (PL_A) as well as the (P_{code}).

The WtSC smart contract triggers an event and the relay node owned by the Wt relays the WtAuthentication certificate to the IoT devices.

The device validates the WtAuthentication certificate then creates and broadcasts its WtAuth packet to the specified IP address of the devices. WtAuth Packet is similar to Eq. (5) but contains the WtAuthentication certificate instead of the Authentication certificate as well as the random generated string of characters which the receiving IoT will hash with the P_{code} from the WtAuthentication certificate. The devices that receive the WtAuth packet sent back its Auth Packet Eq. (8) along with created hashed.

5 Algorithm

ALGORITHM 1 - Smart Contract

1. **Inputs:** The certificate.
2. **Output:** An event is emitted, if required mint NFT and send to appropriate wallet.
3. **if** the certificate is sent from the designated wallet.
4. **then**
5. **if** the certificate is an initialization certificate.
6. **then**
   Return Authentication certificate and IP address of IoT as well as emit event.
7. **else if** Access transfer certificate.
8. **then**
   Return Access grant Certificate and IP address of IoT as well as mint NFT sent to appropriate wallet and emit event.
9. **else** Invalid transaction

6 Value Proposition

The value proposition this mechanism provides is an always available distributed access management and transfer of authorization system. That is interoperable regardless of the vendor of the IoT and the system in which the IoT resides. The paper provides solutions for two scenarios. IoT devices that is not connected directly to the internet; case 1 and case 2 are examples of this scenario. The IoT devices that have access to the blockchain and the internet. The example for this is case 3 and case 4. IoT devices for household are prime examples that one may not want the IoT to have access to the internet. Either for privacy or security reasons as the IoT itself may be vulnerable. The other situation is

that the IoT devices requires dynamic interaction with other IoT devices such as medical devices lend to a patient for monitoring or smart door locks that would need to interact with other wallets.

7 Security Analysis

7.1 The Adversary Mode

The adversary can intercept communication over the network and have the ability to replay, listen in on, delete, or alter any message, as well as create a new message. There are a few assumptions take into consideration, Authenticity of users and devices initially during the Registration phase and that the public key of the device (A_{PK}) is obtained securely initially for the Registration phase as well as the IP address of the devices is static in the network.

7.2 Secret Disclosure Attack

Registration phase: Once the public key of the IoT (A_{PK}) is obtained. The wallet sends the IoT the registration certificate (RG_C) as mentioned in Eq. (1) it encapsulated using the public key of the IoT device this prevent disclosure Attack and the contents of the RG_C remains secure. as the P_{code} is the information that should remain secure.

Device authentication phase: The wallet sends the initialization certificate (IZ_C) to the smart contract any information present in IZ_C other then IP_A is encapsulated with the wallets private key mentioned is Eq. (2). Once the smart contract verifies that transaction is from the wallet that deployed the smart contract. An authentication certificate ($Auth_C$) is created from the IZ_C. The $Auth_C$ uses the IZ_C which is encapsulated with the Private key (W_{IK}) of the wallet mentioned in Eq. (3). This prevents any disclosure Attack. As mentioned before the public (W_{PK}) and private key (W_{IK}) used for the is not the same as the Ethereum public and private key which is used for making transactions.

Device-to-device phase: once the IoT receives the $Auth_C$ it creates a the $AuthPacket_A$ mentioned in Eq. (5) this includes the $Auth_C$ as well a IP_A and Mac_A where the IP_A and Mac_A are encapsulated using the A_{IK} and the $Auth_C$ as mentioned above is encapsulated with the Private key (W_{IK}). Once both the IoT obtains each other's public key by sending each others their *AuthPackets* a secure communication can be established preventing any disclosure Attack.

7.3 Replay Attack

Registration phase: Replay attacks do not pose a problem as the registration certificate (RG_C) as mentioned in Eq. (1) is accept only once and require the RG_C to be encapsulated with the public key of the IoT (A_{PK}). The device would require a reset for the IoT to accept a new registration certificate.

Device authentication phase: The smart contract only accept transaction from the wallet that deployed it on to the blockchain. Authentication certificate ($Auth_C$) being replayed would not pose a threat to the mechanism as there would be no advantage

gained by the adversary. If the adversary sends the $Auth_C$ the IoT device A would check the $Auth_C$ it already has and if they match, The IoT would just discard the new $Auth_C$ send by the adversary. Since the $Auth_C$ is crafted specify for a particular IoT devices it will do no good sending it to other IoT devices on the network, as when the IoT devices does the verification using the MAC address, IP address and the public key as mentioned in Eq. (3) with its own to check whether they match. If they do not the $Auth_C$ is discarded.

Device-to-device phase: The $AuthPacket_A$ as mentioned in Eq. (5) would not pose a threat to replay attacks as it is crafted using the IP_A and Mac_A of the IoT devices A which is then encapsulated using the private key of the IoT device A (A_{IK}). There is no part in the $AuthPacket_A$ that can be tampered. Once the device B receives $AuthPacket_A$ the device sends back its $AuthPacket_B$ mentioned in Eq. (8). The $AuthPacket_B$ includes a random generated string of characters which is hashed with the P_{code} stored in the RG_C and sent back to device B mentioned in Eq. (10). This acts are defense against impersonation attacks.

7.4 Availability

The segregation of the IoT network into different levels allows greater control and prevents the entire network from being compromised. Even if one level of the network was to be compromised the other level could function as normal. Since the Access management is not depended on the vendors availability but rather the blockchain. The mechanism is always available.

8 Conclusion and Future Work

IoT Devices are expected to communicate between devices from different systems in a secure and distributed environment and by different users. In this paper, a distributed and always available authentication and access management mechanism is proposed that permits IoT devices to authenticate between devices and users, regardless of the device vendor. This approach prevents vendor lockdown of IoT devices in a system. Furthermore, the cost of deploying this mechanism is relatively less as the cost is only at the initial stage and at the transfer of access. The device to device and user to devices authentication is accomplished off-chain. The future work would focus on developing a proof of concept to evaluate and test this approach.

References

1. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Decentralized Bus. Rev. 21260 (2008)
2. Novo, O.: Blockchain meets IoT: an architecture for scalable access management in IoT. IEEE Internet Things J. **5**(2), 1184–1195 (2018). https://doi.org/10.1109/JIOT.2018.2812239
3. Szabo, N.: Formalizing and securing relationships on public networks. FM **2**(9) (1997). https://doi.org/10.5210/fm.v2i9.548
4. "EIP-721: Non-Fungible Token Standard". https://eips.ethereum.org/EIPS/eip-721. Accessed 14 Apr 2022

5. Wang, Q., Li, R., Wang, Q., Chen, S.: Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges. arXiv (2021). https://doi.org/10.48550/arxiv.2105.07447
6. Jakobsson, M., Juels, A.: Proofs of work and bread pudding protocols (extended abstract). In: Preneel, B. (ed.) Secure Information Networks. ITIFIP, vol. 23, pp. 258–272. Springer, Boston (1999). https://doi.org/10.1007/978-0-387-35568-9_18
7. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Pap. **151**(2014), 1–32 (2014)
8. He, K., Chen, J., Du, R., Wu, Q., Xue, G., Zhang, X.: DeyPoS: deduplicatable dynamic proof of storage for multi-user environments. IEEE Trans. Comput. **65**(12), 3631–3645 (2016). https://doi.org/10.1109/TC.2016.2560812
9. Bentov, I., Lee, C., Mizrahi, A., Rosenfeld, M.: Proof of activity. SIGMETRICS Perform. Eval. Rev. **42**(3), 34–37 (2014). https://doi.org/10.1145/2695533.2695545
10. Azbeg, K., Ouchetto, O., Jai Andaloussi, S., Fetjah, L.: An overview of blockchain consensus algorithms: comparison, challenges and future directions. In: Saeed, F., Al-Hadhrami, T., Mohammed, F., Mohammed, E. (eds.) Advances on Smart and Soft Computing. AISC, vol. 1188, pp. 357–369. Springer, Singapore (2021). https://doi.org/10.1007/978-981-15-6048-4_31
11. Shahaab, A., Lidgey, B., Hewage, C., Khan, I.: Applicability and appropriateness of distributed ledgers consensus protocols in public and private sectors: a systematic review. IEEE Access **7**, 43622–43636 (2019). https://doi.org/10.1109/ACCESS.2019.2904181
12. Bashar, G., Hill, G., Singha, S., Marella, P., Dagher, G.G., Xiao, J.: Contextualizing consensus protocols in blockchain: a short survey. In: 2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), pp. 190–195 (2019). https://doi.org/10.1109/TPS-ISA48467.2019.00031
13. Li, X., Jiang, P., Chen, T., Luo, X., Wen, Q.: A survey on the security of blockchain systems. Future Gener. Comput. Syst. (2017). https://doi.org/10.1016/j.future.2017.08.020
14. Wu, F., Li, X., Xu, L., Kumari, S., Karuppiah, M., Shen, J.: A lightweight and privacy-preserving mutual authentication scheme for wearable devices assisted by cloud server. Comput. Electr. Eng. (2017). https://doi.org/10.1016/j.compeleceng.2017.04.012
15. Zhang, J., Wang, Z., Yang, Z., Zhang, Q.: Proximity based IoT device authentication. In: IEEE Conference on Computer Communications, IEEE INFOCOM 2017, pp. 1–9 (2017). https://doi.org/10.1109/INFOCOM.2017.8057145
16. Aman, M.N., Chua, K.C., Sikdar, B.: Mutual authentication in IoT systems using physical unclonable functions. IEEE Internet Things J. **4**(5), 1327–1340 (2017). https://doi.org/10.1109/JIOT.2017.2703088
17. Gope, P., Sikdar, B.: Lightweight and privacy-preserving two-factor authentication scheme for IoT devices. IEEE Internet Things J. **6**(1), 580–589 (2019). https://doi.org/10.1109/JIOT.2018.2846299
18. Esfahani, A., et al.: A lightweight authentication mechanism for M2M communications in industrial IoT environment. IEEE Internet Things J. **6**(1), 288–296 (2019). https://doi.org/10.1109/JIOT.2017.2737630
19. Roychoudhury, P., Roychoudhury, B., Saikia, D.K.: Provably secure group authentication and key agreement for machine type communication using Chebyshev's polynomial. Comput. Commun. **127**, 146–157 (2018). https://doi.org/10.1016/j.comcom.2018.06.005
20. Hammi, M.T., Hammi, B., Bellot, P., Serhrouchni, A.: Bubbles of trust: a decentralized blockchain-based authentication system for IoT. Comput. Secur. **78**, 126–142 (2018). https://doi.org/10.1016/j.cose.2018.06.004