PAPER Special Section on Towards Management for Future Communications and Services in Conjunction with Main Topics of APNOMS2022

Secure Enrollment Token Delivery Mechanism for Zero Trust Networks Using Blockchain*

Javier Jose DIAZ RIVERA[†a)], **Waleed AKBAR**[††], **Talha AHMED KHAN**[††], **Afaq MUHAMMAD**[††], *and* **Wang-Cheol SONG**[††], *Nonmembers*

SUMMARY Zero Trust Networking (ZTN) is a security model where no default trust is given to entities in a network infrastructure. The first bastion of security for achieving ZTN is strong identity verification. Several standard methods for assuring a robust identity exist (E.g., OAuth2.0, OpenID Connect). These standards employ JSON Web Tokens (JWT) during the authentication process. However, the use of JWT for One Time Token (OTT) enrollment has a latent security issue. A third party can intercept a JWT, and the payload information can be exposed, revealing the details of the enrollment server. Furthermore, an intercepted JWT could be used for enrollment by an impersonator as long as the JWT remains active. Our proposed mechanism aims to secure the ownership of the OTT by including the JWT as encrypted metadata into a Non-Fungible Token (NFT). The mechanism uses the blockchain Public Key of the intended owner for encrypting the JWT. The blockchain assures the JWT ownership by mapping it to the intended owner's blockchain public address. Our proposed mechanism is applied to an emerging Zero Trust framework (OpenZiti) alongside a permissioned Ethereum blockchain using Hyperledger Besu. The Zero Trust Framework provides enrollment functionality. At the same time, our proposed mechanism based on blockchain and NFT assures the secure distribution of OTTs that is used for the enrollment of identities.

key words: *zero-trust, blockchain, authentication, security, tokens*

1. Introduction

Modern computer networks have evolved outside the boundaries of local infrastructure deployments. Nowadays, the consumption of heterogeneous services with different user requirements is handled by virtualized networks hosted as IaaS by cloud providers. Although cloud computing provides great benefits for tackling the utilization demands required in current network environments, security issues still arise [1], [2]. The line that delimits a network perimeter has dissipated as organizations have a local network infrastructure and remote sites or cloud services for providing access to enterprise resources. Furthermore, current network security models are based on the castle-and-moat concept [3], [4]. This concept considers strict rules and verification for obtaining access from outside the network perimeter, but everyone inside the perimeter is trusted by default. The castle-and-moat method faces several difficulties in modern networks as organizational resources are spread across cloud vendors in different geographical regions, proving a challenge for having a single security model for an entire distributed network.

Zero Trust Networking is a security concept that considers no network entity (e.g., user, device, application, etc.) as trusted [5]–[7]. Zero Trust can be deemed a new paradigm where organizations continuously monitor their network assets, analyze risks, protect, and execute mitigations when necessary [8]. The first bastion of protection involves limiting access to network resources to only identified entities and regularly verifying and authorizing their identity for each subsequent access request. As a result, various technologies, such as OpenID and OAuth2.0, focus on the authentication and authorization process.

Authentication technologies may vary in their methods for handling the identity of the users. However, they commonly use JSON Web Tokens (JWT) [9]–[11] to grant access to resources or provide identity verification. Typically, other forms of identity (e.g., secrets, certificates, signatures) are paired with JWT to assure the ownership of the token. The issue arises for cases where the JWT is used for One Time Token (OTT) enrollment. In an authentication workflow, the enrollment phase establishes the trust between parties, commonly represented by the issuance of a signed certificate (e.g., X.509) from the enroller to the enrollee [12]. Before enrollment, there is no guarantee that the holder of the JWT is the intended enrollee. Techniques such as Multi-Factor Authentication (MFA) or 2-Step verification can use the personal details of the intended enrollee to assure ownership. Still, as OTT has a short period of validity, this added step could induce delays in the enrollment phase, which may render the token invalid.

Another latent issue of the JWT is that the payload, which commonly contains the user ID and (in the case of the OTT) the enrollment server information, can be decoded by any holder of the JWT, raising the concern of a proper delivery mechanism for the enrollment token. In Zero Trust Networking, the actual physical location, IP address/DNS, and the enterprise resources' ports should not be publicly disclosed. A deadlock issue occurs, as the enrollment tokens should not be delivered to an untrusted party. However, an entity can only be trusted if they possess an enrollment token.

This paper is an expanded version of [13] with additional related work discussion and algorithm procedures,

Manuscript received January 10, 2023.
Manuscript revised March 31, 2023.
Manuscript publicized June 1, 2023.

[†]The author is with Department of Electronic Engineering, Jeju National University, Republic of Korea.
[††]The authors are with Department of Computer Engineering, Jeju National University, Republic of Korea.

*This work was first presented at the 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS 2022) and has been revised for this publication.

a) E-mail: ncl@jejunu.ac.kr

DOI: 10.1587/transcom.2022TMP0005

improved performance and stability results, and a comparison between the proposed mechanism and conventional JWT delivery methods. Our mechanism utilizes permissioned blockchain technology and assigns a blockchain identity (i.e., blockchain private key, blockchain public key, and blockchain public address) to each potential enrollee to securely deliver the OTT. The OTT is encrypted using the blockchain public key of the potential enrollee and then included in a Non-Fungible Token (NFT) as part of its tokenURI. The permissioned blockchain ensures that only the intended enrollee can receive and decrypt the token. Our solution does not aim to replace standard authentication mechanisms such as OpenID Connect but rather to enhance their operation by incorporating the blockchain for identity assessment and using NFTs to encode authentication tokens. By doing so, we aim to provide additional security assurances as the encoded JWT can only be accessed and utilized by the intended parties, without relying on a traditional Public Key Infrastructure (PKI). The experimental setup was achieved using a permissioned Ethereum blockchain deployed utilizing the Hyperledger Besu solution. The enrollment functionality is provided by the emerging Zero Trust platform OpenZiti. An RPC server has been integrated inside the OpenZiti platform, where programmatic interaction with the blockchain was added to accomplish our mechanism.

The paper is divided as follows: Section 2 contains the related work, Sect. 3 details the mechanism information, Sect. 4 includes the implementation and results, and Sect. 5 concludes the manuscript.

2. Related Work

Several publications have presented their work on leveraging authentication or authorization using JWT and blockchain technologies. In [14], the authors propose a system that employs a time-based one-time password (TOTP) and two-factor authentication to overcome the security vulnerability in the JWT. The authors referred to the sensitive information included in the token payload, which an attacker could eavesdrop on. The two-factor authentication supplements the use of the JWT as an access token by creating a mirror token as a one-time password (OTP). The OTP token is used instead of the access token for authentication verification. The authors mentioned that the drawback of their proposed system was the increase in the authentication execution time, but no performance metrics were provided.

The authors in [15] propose a blockchain-based smart contract design and implementation for OAuth 2.0 authorization tokens. The scenario presented by the authors considers clients, authorization servers, resource servers, and resource owners. The solution uses bearer tokens in JWT format to specify the authorization requirements for accessing a resource. The authors utilize Ethereum blockchain-based smart contracts to create an ERC-721 (NFT) representation of the base64url encoding of the JWT. The NFT provides proof of possession of the bearer token as it can be mapped directly to the public blockchain address of the intended owner. The authors mention security assumptions that should be in place to guarantee the identity of resource servers and clients (e.g., X.509 certificates). The authors present performance and cost evaluation, where details of the blockchain transaction costs are shown in a table. Furthermore, there is mention of computational overhead related to the invocation of smart contract functions, but no metrics or comparisons with standard OAuth 2.0 authentication methods are provided.

The authors in [16] tackle decentralized authentication by proposing a mechanism named AuthChain, which relies on blockchain functionality to provide a robust, transparent, and secure method for user authentication. The motivation of the research is centered on the latent security implications of credential information storage in current authentication systems, which may be susceptible to account hacking and data breaches. To this end, blockchain technology is utilized for decentralized authentication employing smart contract operation. The authentication process is established as a self-governed process programmed through a verifiable set of rules. The mechanism leverage the functionality of blockchain cryptography to hash and store the information in the blockchain for verification. The benefits presented by the authors rely on defense from man-in-the-middle attacks, as possible intruders cannot generate valid verification hashes due to not possessing the Public and Private keys used for the hash function. Although the disclosed mechanism provides a reasonable solution for decentralized authentication, the implementation relies on a permanent hash record for continuous verification. This means that the blockchain does not reflect the current state of the registered user, as the authentication system takes for granted that once a hash is registered in the blockchain, it can trust any authentication request that matches the hash. In contrast, a more dynamic solution can leverage tokens that are generated, utilized, and discarded once the authentication process is finished. The tokens can guarantee that every authentication request is unique for every point in time. The paper makes mention of tokens as unique identifiers, but the authors do not elaborate on their actual use for the proposed mechanism.

In [17], the authors proposed a blockchain-based authentication and access control model for smart environments. The proposed model used a hybrid blockchain consisting of a public Ethereum network connected to multiple private Hyperledger fabric networks. The public blockchain serves as integrated authentication management where users can create their identity with their Ethereum account information (blockchain public address), hashed secret, and personal information. A smart contract handles the creation of the identity. The multiple private blockchain networks represent different organizations where the integrated ID can be used to authenticate and access organization resources. Only identities that have previously been registered from the public blockchain network and verified by the private blockchain can access specific resources from an organization. The model proposed in this paper provides a tamper-proof record of access history related to a specific ID. However, due to the use of a public Ethereum network, creating an integrated

ID and registering the organization Token for a specific ID can incur high transaction fees that the user must cover. The authors do not mention any drawbacks related to the transaction costs. Furthermore, the validation token distribution mechanism is not fully described, a reader may assume that the token is transferred using blockchain methods, but no reference to blockchain token standards (e.g., ERC-21 or ERC-721) is disclosed in the paper. Also, any user (authorized or not) can interact with the identity smart contract and issue the creation of an integrated ID.

The proposed work in the present manuscript takes the ERC-721 approach of [15] as its basis but applies it to authentication instead of authorization. Our research aims to provide a secure distribution mechanism for enrollment tokens to create a strong identity. As mentioned in the related work of this section, JWT's base64url encoding is by no means a secure encryption. A mechanism should ensure that the holder of a JWT has the rights to it. No other entity should be able to decode the JWT, as any information included in the payload could be sensible for revealing the network location of enterprise assets. Nonetheless, JWT has become a common approach in many systems requiring access control. Due to this, our approach aims to securely define the creation and distribution of JWT that serve as OTT for identity enrollment to achieve the requirements of Zero Trust networking.

This manuscript utilizes a permissioned Ethereum blockchain alongside the OpenZiti platform to apply the proposed mechanism for secure OTT delivery. The main contributions of this paper are as follows:

- A secure mechanism for assuring tamper-proof of the JWT payload based on blockchain Public key encryption of the JWT's base64url.
- Identity management is based on permissioned blockchain records where only authorized blockchain public addresses are allowed to participate.
- Applying the ERC-721 token standard for NFT creation where the token metadata includes the encrypted JWT's base64url as part of the "onchain" information.
- The blockchain assures the proper delivery of the JWT as the NFT provides a "proof-of-ownership" related to mapping the intended enrollee's public blockchain address to the owner field of the NFT.
- An immutable ledger where every transaction can be verified and audited by an authorized entity.

3. Mechanism Description

The design for our proposed mechanism can be seen in Fig. 1. It is representative of a Zero Trust solution where no entity is considered trusted by default requiring a verified identity for connecting to the network infrastructure. All communications are based on digitally signed messages to prevent eavesdropping or impersonation. The trustworthiness of each component is constantly verified to ensure security.

The system's main components are *a permissioned blockchain*, *an RPC server*, *an enrollment server* (as part of a secure network overlay), and *endpoints* representing clients or service providers that require enrollment into the network. The *enrollment server* and *RPC server* require their own private and public keys (PrK, PuK) to interact with the blockchain, and a trustworthy administrator entity should manage their initial setup. The *blockchain* verifies trust and ensures that no malicious entities can compromise the system. The external *endpoints* are subject to the authentication mechanism described in this manuscript. Impersonation cannot occur unless the private keys associated with a blockchain account are exposed, ensuring that only authorized entities are able to access the system.

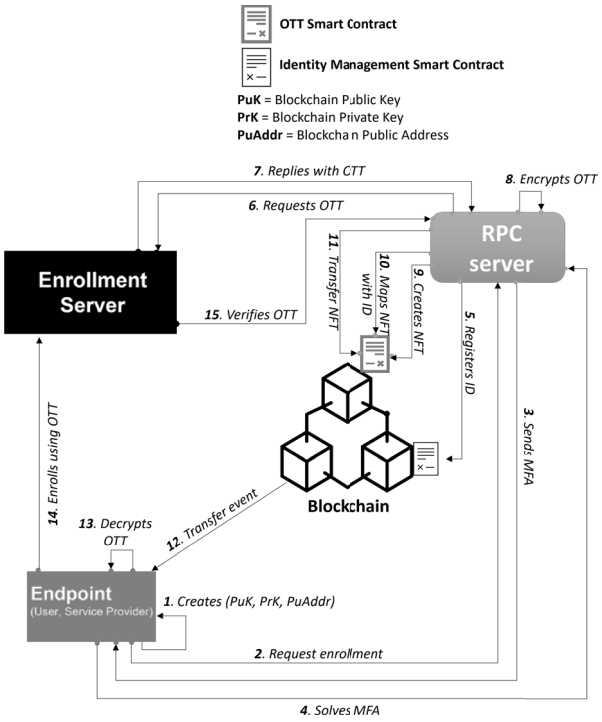

Fig. 1 Mechanism for secure OTT delivery.

In the context of standard authentication mechanisms such as OpenID Connect or Security Assertion Markup Language (SAML), the Identity Provider (IdP) is responsible for authenticating the user and providing an identity token to the relying party or client application. The *enrollment server, blockchain,* and *RPC server* in the discussed mechanism can be seen as a modified version of an IdP, where the traditional identity verification process leverages the security guarantees provided by the blockchain.

3.1 Permissioned Blockchain

Blockchain technology creates a data structure with built-in security features. It is based on encryption, decentralization, and consensus principles, ensuring transaction confidence. Most blockchains or distributed ledger technology (DLT) data is organized into blocks containing one or more transactions. Each new block in a cryptographic chain connects

Table 1 Identity struct.

registerIdentity(Address PuAddr, string name, string e-mail)	
PuAddr	The blockchain public address of the requesting enrollee
name	Plain text name of the requesting enrollee
e-mail	The e-mail of the requesting enrollee

to all the blocks before it, so tampering is nearly impossible. A consensus mechanism validates and agrees on all transactions within the blocks, ensuring that each transaction is truthful and correct [18].

Most popular blockchains operate as a public ledger where any entity can participate either as part of the consensus or as a blockchain client that executes transactions. Furthermore, the blockchain technology that provides a programmatic framework has allowed the creation of decentralized applications that can accommodate various use cases [19]. However, transaction costs must be carefully considered if businesses or organizations desire to use this technology [20].

Private blockchains provide the benefits of public blockchains plus functionality that can accommodate specific business rules of different organizations and enterprise scenarios [21], [22]. Due to this, our solution considers the use of a private/permissioned blockchain capable of programmatic interactions through the use of smart contracts.

The blockchain functionality required for our mechanism is defined in 2 smart contracts, one for *identity management* and another for *OTT issuance* - ERC-721 (NFT). The *identity management* smart contract handles the registration of blockchain public addresses (PuAddr) of entities identified as part of an organization/enterprise by using the method *registerIdentity()*. The information is handled as a struct with the attributes seen in Table 1. The *OTT issuance* smart contract is used to create the NFT that holds the encrypted JWT information as part of the tokenURI.

3.2 Enrollment Server

The enrollment server has two main functionalities. First, it is in charge of generating the OTT for the verified identities that exist in the blockchain. Second, it provides enrollment of identities into the Zero Trust network by signing and provisioning X.509 certificates. These certificates are used for mutual Transport Layer Security (mTLS) connections inside the Zero Trust network. The scope of this research paper focuses on the first functionality (i.e., OTT creation). If no verification method for the handler of the OTT is in place, any entity that holds the token could trigger an enrollment procedure and generate valid certificates that would grant further entrance into the Zero Trust Network.

The enrollment server is considered part of the Zero Trust network overlay. In our scenario, the enrollment capabilities are provided by the OpenZiti framework developed by Netfoundry [23]. The components of this framework are the *ziti controller* and *ziti routers (edge and fabric)* that are interconnected to form a secure overlay network that provides end-to-end encryption between dark services and ziti clients. The *enrollment server* functionalities for identity creation, OTT generation, and identity enrollment are inside the *ziti controller*. By default, an authorized administrator user can interact with the *ziti controller* to manually create identities (e.g., CLI commands, API calls) and generate the JWT used for OTT. However, OpenZiti has no integrated mechanism for secure delivery of the generated JWT to the intended entity. An example of the plaintext contents of a JWT used for OTT can be seen in Fig. 2.

```
{
"em": "ott",
"exp": 1647917167,
"iss": "https://ziti-edge-controller:1280}",
"jti": "716a8f15-5bf5-40dc-9320-1d2857b7759b",
"sub": "KIv6CJRUv"
}
```

Fig. 2 JWT payload contents.

3.3 RPC Server

This server is the entity that handles the user requests for identity creation, the execution of the smart contracts, and the interactions with the *enrollment server* for secure OTT creation. In essence, this server verifies that requesting entities are part of the organization/enterprise to register them in the blockchain. The RPC server must be aware of the business rules and human resource information particular to an organization. This work assumes that the RPC server has these requirements fulfilled to apply a multi-factor authentication (MFA) method based on verified personal contact information.

When the users solve the MFA, the RPC server obtains a cryptographically signed request. This signature is used to obtain the PuK and the PuAddr of the requesting user. The PuAddr is used to register the identity utilizing the *identity management* smart contract, and the PuK is used for the encryption of the JWT. The process of signing and recovering the blockchain PuAddr and PuK is shown in Algorithm 1.

The RPC server operates as a decentralized application (dapp) that can combine the information obtained from the methods of the *identity management* and the *OTT issuance*

Algorithm 1 Obtaining PuAddr and PuK from Signature

Input: M - MFA message
PrK - Blockchain private key of the endpoint
Output: $PuAddr$- Blockchain public address of the endpoint
PuK - Blockchain Public Key of the endpoint

procedure ENDPOINT SIGNS MESSAGE M
  hashedM ← *hash.keccak256*(M)
  signedM ← *sign*(*PrK*, *hashedM*)
  solveMFA(hashedM, signedM)
procedure RPC SERVER- RECOVERS SIGNER PUADDR AND PUK
  PuAddr ← *recover*(*signedM, hashedM*)
  PuK ← *recoverPublicKey*(*signedM, hashedM*)

Note:
solveMFA(() - sends $hashedM$ and $signedM$ to RPC server
sign(), hash(), recover(), recoverPublicKey() - part of *eth-crypto* library

Algorithm 2 Secure OTT Delivery

Input: $PuAddr$ - Blockchain public address of the endpoint
PuK - Blockchain Public Key of the endpoint
Output: OTT- Decrypted JWT ready to use for enrollment

procedure RPC SERVER - OTT CREATION AND ENCRYPTION
  OTT ← *requestOTT*($PuAddr$)
  expOTT, jtiOTT ← *extractMetadata*(*OTT*)
  encryptedOTT ← *encryptWithPublicKey*(PuK, OTT)
  nftOTT ← *createNFT(encryptedOTT)*
  mapOTTinfo(tokenId, expOTT, jtiOTT)
  transfer(tokenId, PuAddr)
procedure ENDPOINT - RECEIVES $nftOTT$ AND DECRYPTS
  encryptedOTT ← *extractTokenURI*(*tokenId*)
  OTT ← *decryptWithPrivateKey*(*PrK, encryptedOTT*)

Note:
requestOTT() - from RPC server to enrollment server
encryptWithPublicKey() - part of *eth-crypto* library
decryptWithPrivateKey() - part of *eth-crypto* library
requestOTT() - enrollment server replies to RPC server with OTT
expOTT - expiration time of OTT
jtiOTT - JWT identifier of OTT
mapOTTInfo() - Relationship between NFTs and OTT metadata
tokenId - Identifier of created NFT - $nftOTT$
PrK - Blockchain Private Key of the endpoint

smart contracts. The mapping of the registered blockchain PuAddr inside the identities struct is used to verify the ownership of the OTT and its status (e.g., active, expired).

The RPC server executes OTT creation requests to the *enrollment server* for verified identities (i.e., registered in the blockchain). This procedure requires the RPC server to have the proper credentials for interaction. Our approach considers the RPC server as an administrator type of user authorized to execute API calls to the *enrollment server*, which automatically replies with a JWT representing the OTT used for enrolling. The OTT encryption uses the blockchain PuK of the verified identities to guarantee that only the intended owner of the token can utilize it for enrollment. The encrypted data is included as part of NFT metadata by interaction with the *OTT issuance* smart contract. Furthermore, the generated NFT *tokenId* is associated with the "jti" and "exp" information of the JWT as part of the *OTT issuance* smart contract functionality. Algorithm 2 showcases the process of secure delivery of the OTT.

3.4 Endpoints

The endpoints represent the clients and service providers that request identity creation and enrollment into the Zero Trust network. Every endpoint is considered a blockchain client that holds blockchain PrK and PuK alongside a blockchain PuAddr. An endpoint is unaware of the *enrollment server* until it receives the encrypted OTT. Initially, the endpoints request to create an identity by interaction with the RPC server that triggers an MFA procedure. The solution of the MFA includes a call to a *setApprovalForAll()* method that delegates the transfer of NFTs to the RPC server as follows.

$$setApprovalForAll(operator, approved) \quad (1)$$

The method in Eq. (1) allows an operator to manage any NFT held by the endpoint that initiated the call to the method. The *operator* is a string that refers to the PuAddr of the RPC server, and the *approved* argument is a boolean that needs to be set to True. When an endpoint receives an encrypted OTT as an NFT, it initiates the enrollment procedure after decryption of the tokenURI using its blockchain PrK. This process triggers a state verification of the OTT by the RPC server, which will "burn" the endpoint NFT as soon as the OTT is used for enrollment or if the token has an "exp" date already passed. The NFT "burn" is triggered by the method *transferFrom()*, as seen in Eq. (2).

$$transferFrom(endpoint, genesis, tokenId) \quad (2)$$

Algorithm 3 Token verification and burning

Input: OTT - decrypted One Time Token
$signedEM$ - PrK signed jti of OTT
Output: $transferFrom()$- Burning of token

procedure ENDPOINT - ENROLLMENT REQUEST
  enrollRequest(signedEM, OTT)
procedure ENROLLMENT SERVER - PROCESS ENROLLMENT
  isValid ← *isTokenValid(signedEM, OTT)*
  if *isValid* **then**
    ottUsed(OTT)
procedure RPC SERVER - VALIDATE OTT
  jtiOTT ← *extractJTI*(*OTT*)
  expOTT, tokenId ← *getOTTInfo*(*jtiOTT*)
  ownerPuAddr ← *ownerOf*(*tokenId*)
  if *expOTT* < block.timestamp **then**
    burn(ownerPuAddr,tokenId)
    return $False$
  recPuAddr ← *recover*(*jtiOTT, signedEM*)
  if *recPuAddr* ! = *ownerPuAddr* **then**
    burn(ownerPuAddr,tokenId)
    return $False$
  return $True$
procedure RPC SERVER - HANDLE USED OTT
  jtiOTT ← *extractJTI*(*OTT*)
  tokenId ← *getOTTInfo*(*jtiOTT*)
  ownerPuAddr ← *ownerOf*(*tokenId*)
  burn(ownerPuAddr,tokenId)
procedure RPC SERVER - BURN OTT
  transferFrom(ownerPuAddr,genesis,tokenId)

Note:
enrollRequest() - from endpoint to enrollment server
isTokenValid() - OTT validation request to the RPC server
ottUsed() - from enrollment server to RPC server for burning a used token
extractJTI() - extracts JWT identifier from OTT
getOTTInfo() - obtains the expiration time and the blockchain tokenId
expOTT - expiration time of OTT
jtiOTT - JWT identifier of OTT
tokenId - identifier of the NFT - (OTT in the blockchain)
ownerOf() - obtains the blockchain PuAddr that owns the OTT
burn() - call to the burn procedure
recover() - obtains the PuAddr of the endpoint that is using the OTT
transferFrom() - transfers the token to the genesis address
PrK - Blockchain Private Key of the endpoint
genesis - 0x0000000000000000000000000000000000000000 address
ownerPuAddr - PuAddr that owns the OTT as an NFT
recPuAddr - PuAddr from the signed enrollment request - *signedEM*

The RPC server executes the *transferFrom()* method to

transfer the expired/used OTT (specified by a tokenId) from an endpoint address to the genesis address. In a blockchain, it is a common practice to execute a token "burn" by sending them to the *"0x0000000· · ·0000000"* (i.e., genesis) address. Algorithm 3 details the process that renders the tokens unusable, effectively removing them from the scenario.

4. Implementation and Results

To illustrate the operation of the proposed mechanism, a private Ethereum blockchain was utilized. The deployment was achieved by using the Hyperledger Besu blockchain [24] implementation, which provides configuration for creating a permissioned blockchain that supports a Proof-of-Authority (PoA) consensus mechanism [25], [26]. In an ideal scenario, an organization can identify and select trusted nodes that may serve as validators inside the blockchain network. For the results presented in this manuscript, a blockchain consisting of 5 nodes utilizing the Clique consensus mechanism was configured. This consensus mechanism does not require high computing power to produce and verify blocks. Furthermore, the gas fee for the blockchain transactions was configured with a value of "zero" to achieve a *no-fee* blockchain network in order to exclude the transaction costs for this private network.

4.1 Test-Bed Preparation

The identity management and *OTT issuance* smart contracts have been coded using the solidity programming language and deployed into the blockchain using the remix IDE. The RPC server is a dapp developed using NodeJS and javascript. It utilizes the *eth-crypto* [27] libraries to create its own blockchain PrK, PuK, and PuAddr. The PuAddr of the RPC server has been configured as the *owner()* of the deployed smart contracts. Except for the *setApprovalForAll()*, all the smart contract functions that generate a blockchain transaction were modified only to accept calls by the owner.

A simple NodeJS app alongside Metamask was used to illustrate the interaction that endpoints (representing clients/service providers) can execute to identify themselves in the Zero Trust Network. The NodeJS app utilizes the eth-crypto libraries for the decryption of the NFT metadata. Metamask handles the creation of the blockchain PrK, PuK, and the PuAddr of the client. Multiple clients can be identified by their unique blockchain PuAddr.

OpenZiti was deployed as the Zero Trust network overlay. The enrollment functionalities of the *ziti-controller*, namely the JWT generation and OTT enrollment, have been identified and considered as the *enrollment server* for the experiments. A Ziti administrator account for API interaction was created and provided to the RPC server. Minimal modification of the golang code for the ziti-controller was done. Additional steps for verification of the OTT owner and notification of the status of the enrollment token (e.g., used, expired) have been added.

A simple e-mail-based MFA authentication has been integrated with the RPC server. It utilizes predefined e-mail addresses that serve as the "verified" contact information of the endpoints that request enrollment into the Zero Trust Network. The contents of the e-mail point to the dapp functionality for signing the verification message and triggering the *setApprovalForAll()* method.

4.2 Experiment

To verify the proposed mechanism, the following experiment was conducted, which includes the numbered steps illustrated in Fig. 1.

1. A user creates its PrK, PuK, and PuAddr by using Metamask.
2. The user interacts with the RPC server and requests enrollment by providing the following information:
  a) PuAddr, Name, E-mail.
3. The RPC server verifies the information.
  a) If the e-mail matches the organizational e-mail from a database (Simplified in our scenario). It issues an MFA verification.
  b) Else, it denies the request.
4. The user opens a verification link from the e-mail.
  a) Uses *eth-crypto* to sign a response to the RPC server using its PrK.
  b) The verification link includes a call to *setApprovalForAll()* and delegates NFT transfers to the RPC server.
5. The RPC server registers the user information inside the blockchain using the identity management smart contract.
  a) PuAddr, Name, E-mail.
6. The RPC server communicates with the enrollment server for the creation of an OTT (JWT) for the registered user.
  a) Ziti admin APIs are used.
7. The enrollment server issues an OTT in JWT format as a reply to the RPC server.
8. The RPC server executes the following:
  a) Extracts the "exp" and "jti" fields of the JWT.
  b) Encrypts the JWT using the PuK of the user (Obtained from the signature in step 4.a).
9. The RPC server creates an NFT by interaction with the OTT issuance smart contract. The tokenURI of the NFT contains the encrypted JWT.
10. The RPC server executes the *mapOTTinfo()* method from the OTT issuance smart contract. It inputs the following information.
  a) tokenId, (of the created NFT from step 9, "jti" and "exp" (from step 8.a).
11. The RPC server executes the *transfer()* method in the OTT issuance smart contract, sending the NFT to the PuAddr obtained in step 2. A smart contract event for the transfer is triggered. The event contains the following information.
  a) sender PuAddr, recipient PuAddr, tokenId.
12. The user is notified of the transfer (by the event) and

queries the NFT's tokenURI by the tokenId.

13. The user decrypts the JWT information using its PrK.

14. The user utilizes the decrypted information as an OTT for enrollment by communicating with the enrollment server.
   a) It uses the native OpenZiti client API functionality for enrollment.
   b) It appends a PrK signed message to the request.

15. The enrollment server verifies the OTT validity and ownership by interaction with the RPC server. *isTokenValid()* receives the OTT in JWT format and a signed message to verify the blockchain information relative to the OTT.
   a) It uses the "jti" to query the "exp" field, tokenId, and PuAddr that owns the tokenId registered in the blockchain.
   b) If "exp" is less than the current timestamp, the OTT is invalid.
   c) It recovers the user PuAddr from the signed message in step 14.b. If the recovered PuAddr (*recPuAddr*) is not equal to owner PuAddr (*ownerPuAddr*) the OTT is invalid.
   d) Invalid tokens are burned as per Eq. (2).

The enrollment server uses a valid OTT to complete the enrollment procedure (not discussed in the manuscript). It sends a notification to the RPC Server regarding the status of the OTT using the *ottUsed()* method. The tokens that were used are burned in accordance with Eq. (2). The identity of the enrollee and the validity of the OTT are verified and ensured throughout the entire enrollment process.

4.3 Results

The performance of the secure token delivery mechanism was evaluated based on the total completion time (T_{C_t}) of the enrollment process. The criterion of the evaluation is represented by Eq. (3). Where for an iteration i, T_{β_t} is the total transaction block time in milliseconds, T_{ε_t} is the total signature, encryption, and decryption-related tasks time in milliseconds, T_{Q_t} is the total blockchain query time and T_{μ_t} is the total time of tasks that are not measured with a near-constant value of time (e.g., MFA verification, e-mail, manual intervention, etc.).

$$T_{C_t} = \frac{\sum_{i=0}^{n}(T_{\beta_{t_i}} + T_{\varepsilon_{t_i}} + T_{Q_{t_i}} + T_{\mu_{t_i}})}{n} \tag{3}$$

Figure 3 displays the average results in seconds for 50 iterations of the enrollment process for different block-time configurations (e.g., 2 seconds, 5 seconds, 15 seconds). The results show that the T_{β_t} plays a key role in improving the performance of the proposed enrollment process. Nonetheless, a lower block time might reach a diminishing improvement as the total competition time gets affected by the T_{μ_t}, which includes tasks that have high variance in time for different iterations (not constant). As per our experiments, the encryption, decryption, and signature tasks do not greatly affect the enrollment process; each task has an average of 20 milliseconds to complete.

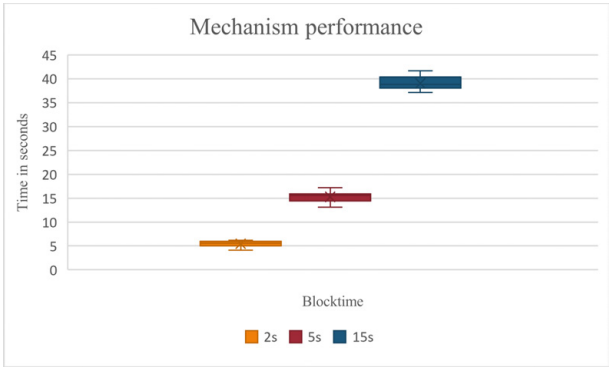

Fig. 3 Mechanism's performance measured in seconds.

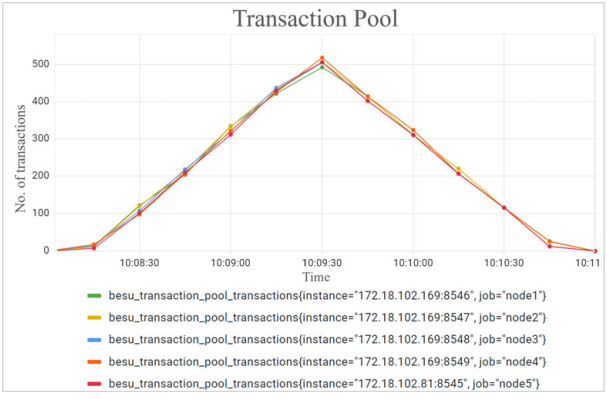

Fig. 4 Stability of the blockchain system.

Figure 4 illustrates the results of the test conducted in order to assess the stability of the system. The configured block time (β_t) for the test was 2 s with a total of 1000 OTT $transfer()$ transactions executed simultaneously using a multi-threaded python environment. The blockchain system can handle a load of transactions employing a transaction pool per node that acts as a queue of unprocessed transactions. The distribution of transactions per node is uniform which denotes good node synchronization and immediate block propagation.

The best performance achieved by the proposed mechanism for delivering an OTT to a requesting enrollee is 3600 ms when considering a β_t of 2 s (not including the burning step). The standard operation (base OpenZiti without blockchain) of token delivery using a JWT as an OTT completes in approximately 100 ms. There is still a great margin for improvement of our mechanism in order to compete with standard JWT implementations when considering only the performance requirements. Nonetheless, the base JWT delivery utilized in OpenZiti can be a target for security breaches where an attacker can intercept, decode and utilize the JWT before the intended enrollee. The main benefits our mechanism brings to the security of the JWT delivery are disclosed in Table 2, which are achieved by the encryption

Table 2 Base JWT delivery vs proposed mechanism.

Scheme	Hidden Payload	Ownership Verification	Revocation
OpenZiti - JWT as OTT	not provided	not provided	not provided
Proposed Mechanism	provided	provided	provided

of the JWT payload, the mapping proof-of-ownership of the JWT as NFT, and the revocation capabilities given by the smart contract functionality.

5. Conclusion

A mechanism for secure enrollment token distribution for Zero Trust networks using blockchain was presented in this manuscript. The fundamental requisite of securing a strong identity for Zero Trust is realized through blockchain techniques such as non-fungible tokens (NFT). The proof of ownership that NFT provides allows for a trustable verification of potential enrollees into the system. Furthermore, the protection of the enrollment information is achieved by encrypting the metadata of the NFT. Without blockchain technology, a centralized approach that requires a complex PKI infrastructure and database management would be required to achieve similar results. Even though the proposed mechanism follows the principles of Zero Trust ("never trust, always verify"), it is important to implement appropriate measures to protect the blockchain private keys of each component to prevent breaches that can compromise the network system's security.

The private permissioned blockchain can be tailored to match the business rules of specific organizations with more flexible approaches compared to the public blockchain implementations (e.g., block time, transaction fee, number of validators, consensus algorithm). Nonetheless, planning is required to achieve the mechanism's best performance. Reducing the block does not necessarily improve the mechanism's completion time and could lead to issues such as breaking the synchronization of block validators. (Depending on the size and load of the blockchain). As blockchain technology matures, the performance of our mechanism can improve and close the gap of the performance margin between baseline JWT technologies.

The proposed mechanism can potentially be applied to other systems that utilize authentication and authorization mechanisms similar to OpenID Connect, SAML, or OAuth2.0 by replacing the traditional Identity Provider functionality with the *enrollment server, permissioned blockchain,* and *RPC server* components. However, the specific implementation and integration of the proposed mechanism would depend on the requirements and architecture of each particular system.

Currently, the RPC server's role is to operate the smart contracts' functionality and forward requests from the endpoints to the enrollment server. Extended work could improve the operation of the RPC into a more intelligent approach that handles the smart control of access policies related to the use of the Zero Trust network resources. This requires a more complex system where real-time monitoring and analysis of the operation state of all the resources that form part of the Zero Trust network is considered. Further research is oriented in this direction.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B01016322).

References

[1] I. Odun-Ayo, O. Ajayi, and S. Misra, "Cloud computing security: Issues and developments," Proc. World Congress on Engineering 2018 (WCE), London, U.K., vol.1, pp.175–181, July 2018.

[2] G. Kumar, "A review on data protection of cloud computing security, benefits, risks and suggestions," United International Journal for Research & Technology (UIJRT), vol.1, no.2, pp.26–34, 2019.

[3] A. Deshpande, "A study on rapid adoption of zero trust network architectures by global organizations due to COVID-19 pandemic," New Visions in Science and Technology, vol.1, ed. S.M. Lawan, pp.26–33, B P International, India, 2021.

[4] G. Sorrentino, "The human variable: Designing a security strategy for a future in flux," Cyber Security: A Peer-Reviewed Journal, vol.5, no.1, pp.6–12, 2021.

[5] A. Deshpande, "Relevance of zero trust network architecture amidts and it's rapid adoption amidts work from home enforced by COVID-19," Psychology and Education Journal, vol.58, no.1, pp.5672–5677, 2021.

[6] P. Assunção, "A zero trust approach to network security," Proc. Digital Privacy and Security Conference 2019, Porto, Portugal, pp.65–72, Jan. 2019.

[7] E. Bertino, "Zero trust architecture: Does it help?," IEEE Security Privacy, vol.19, no.5, pp.95–96, 2021.

[8] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," NIST Special Publication 800-207, National Institute of Standards and Technology, U.S.A. 2020. DOI: 10.6028/NIST.SP.800-207

[9] Y. Sadqi, Y. Belfaik, and S. Safi, "Web oauth-based SSO systems security," Proc. 3rd International Conference on Networking, Information Systems & Security (NISS), Marrakech, Morocco, pp.1–7, March 2020.

[10] R. Watt, "Proof-of-possession tokens in microservice architectures," Master's thesis, Department of Computing Science, Faculty of Science and Technology, Umeå University, pp.1–43, 2018.

[11] I.P.A. Pratama Linawait, and N.P. Sastra, "Token-based single sign-on with JWT as information system dashboard for government," Telecommunication, Computing, Electronics and Control (TELKOMNIKA), vol.16, no.4, pp.1745–1751, 2018. DOI: 10.12928/TELKOMNIKA.v16i4.8388

[12] T.T. Mini, "Secure device identifiers and device enrollment in industrial control system," Proc. 13th IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Goa, India, pp.1–5, Dec. 2019. DOI: 10.1109/ANTS47819.2019.9118131

[13] J.J. Diaz Rivera, T.A. Khan, W. Akbar, A. Muhammad, and W.C. Song, "Secure enrollment token delivery for Zero Trust networks using blockchain," Proc. 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS), Takamatsu, Japan, pp.1–6, Sept. 2022. DOI: 10.23919/APNOMS56106.2022.9919940.

[14] W.S. Park, D.Y. Hwang, and K.H. Kim, "A TOTP-based two factor authentication scheme for hyperledger fabric blockchain," Proc. Tenth International Conference on Ubiquitous and Future Networks (ICUFN), Prague, Czech Republic, pp.817–819, July 2018. DOI: 10.1109/ICUFN.2018.8436784
[15] N. Fotiou, I. Pittaras, V.A. Siris, S. Voulgaris, and G.C. Polyzos, "OAuth 2.0 authorization using blockchain-based tokens," Proc. Workshop on Decentralized IoT Systems and Security (DISS), San Diego, U.S.A., pp.1–6, Feb. 2020. DOI: 10.14722/diss.2020.23002
[16] S.Y. Lim, P.T. Fotsing, O. Musa, and A. Almasri, "AuthChain: A decentralized blockchain-based authentication system," International Journal of Engineering Trends and Technology (IJETT), pp.70–74, 2020. DOI: 10.14445/22315381/CATI1P212
[17] N. Choi and H. Kim, "A novel blockchain-based authentication and access control model for smart environment," Advances in Science, Technology and Engineering Systems Journal (ASTES), vol.6, no.1, pp.651–657, 2021. DOI: 10.25046/aj060171
[18] M. Smits and J. Hulstijn, "Blockchain applications and institutional trust," Front. Blockchain, vol.3, article 5, pp.1–13, 2020. DOI: 10.3389/fbloc.2020.00005
[19] W. Cai, Z. Wang, J.B. Ernst, Z. Hong, C. Feng, and V.C.M. Leung, "Decentralized applications: The blockchain-empowered software system," IEEE Access, vol.6, pp.53019–53033, 2018. DOI: 10.1109/ACCESS.2018.2870644
[20] D. Meva, "Issues and challenges with blockchain: A survey," International Journal of Computer Sciences and Engineering (IJCSE), vol.6, no.12, pp.488–491, 2018. DOI: 10.26438/ijcse/v6i12.488491
[21] R. Yang, R. Wakefield, S. Lyu, S. Jayasuriya, F. Han, X. Yi, X. Yang, G. Amarasinghe, and S. Chen, "Public and private blockchain in construction business process and information integration," Automation in Construction, vol.118, pp.1–21, 2020. DOI: 10.1016/j.autcon.2020.103276
[22] Y. Hao, Y. Li, Z. Dong, L. Fang, and P. Chen, "Performance analysis of consensus algorithm in private blockchain," Proc. IEEE Symposium on Intelligent Vehicles, Changshu, China, pp.280–285, June 2018. DOI: 10.1109/IVS.2018.8500557
[23] OpenZiti, Netfoundry. [Online], https://openziti.github.io/, accessed Jan. 3. 2023.
[24] Hyperledger Besu, Hyperledger Foundation. [Online], https://www.hyperledger.org/use/besu, accessed Jan. 3. 2023.
[25] S.D. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "PBFT vs proof-of-authority: applying the cap theorem to permissioned blockchain," Proc. Italian Conference on Cybersecurity, Venice, Italy, pp.1–11, Jan. 2017.
[26] N.A. Asad, T. Elahi, A.A. Hasan, and M.A. Yousuf, "Permission-based blockchain with proof of authority for secured healthcare data sharing," Proc. 2nd International Conference on Advanced Information and Communication Technology (ICAICT), Dhaka, Bangladesh, pp.35–40, Nov. 2020. DOI: 10.1109/ICAICT51780.2020.9333488
[27] eth-crypto, Daniel Meyer. [Online], https://github.com/pubkey/eth-crypto, accessed Jan. 3. 2023.

Javier Jose Diaz Rivera received his B.S. in Computer Systems Engineering from Monterrey Institute of Technology and Higher Education (Mexico) in 2005, and M.S. degree in Computer Engineering from Jeju National University (Korea) in 2019. He is currently pursuing a doctorate in Electronics Engineering at Jeju National University. His research interests include SDN, NFV, Blockchain technology, and network security.

Waleed Akbar received his B.S. in Telecommunication and Networking in 2014 and his M.S. in Computer Science in 2018 from the Abbottabad campus of COMSATS University (Pakistan). He received his Ph.D. in Computer Engineering from Jeju National University (Korea) in 2022. He is currently working as a lecturer at COMSATS University, Pakistan. His research interests include SDN, NFV, big data processing, network data analytics, and network monitoring.

Talha Ahmed Khan received his B.S. degree in Computer Science from FAST- National University of Computer and Emerging Sciences (Pakistan). He received his M.S. and Ph.D. degrees in Computer Engineering in 2019 and 2022, respectively, from Jeju National University, (Korea). His research interests include the SDN, NFV, 5G Mobile Networks, Intent-based Networking, Network Orchestration, Mobile Edge Computing and VNF.

Afaq Muhammad received a B.S. degree in Electrical Engineering from the University of Eng. and Technology (Pakistan) in 2007. He received a MS degree in Electrical Engineering with an emphasis on Telecom from Blekinge Institute of Technology, (Sweden) in 2010, and a Ph.D. degree in Computer Engineering from Jeju National University (Korea) in 2017. Currently, he is working as a Postdoctoral Researcher at Network Convergence Lab, Jeju National University. His research interests are cloud computing, SDN, NFV, wireless networks and protocols, machine learning, and data science.

Wang-Cheol Song received a B.S. degree in Food Engineering from Yonsei University (Korea) in 1986. He received B.S., M.S., and Ph.D. in Electronics from Yonsei University, in 1989, 1991 and 1995, respectively. He has worked as a full professor at the Department of Computer Engineering, Jeju National University, Korea, Since 1996. His research interests include VANETs and MANETs, SDN/NFV, IBN, and network management.