The Future of Digital Authentication: Blockchain-driven Decentralized Authentication in Web 3.0

Jungwon Seo

Department of Computer Science and Engineering, Sogang University, 915 Ricci Hall 35 Baekbeom-Ro, Mapo-gu, Seoul, South Korea
E-mail: jungwon@sogang.ac.kr

Received 22 January 2024; Accepted 19 June 2024

Abstract

This paper presents an innovative Web 3.0 authentication technique, designed for a user-centric internet environment. Addressing the rising demand for authentication techniques suitable for Web 3.0, it defines the essential features of such systems and introduces a new approach using smart contracts. This approach utilizes mother and child tokens in conjunction with the lock smart contract to ensure secure authentication. The approach is thoroughly tested against various security threats, including man-in-the-middle, replay, and brute-force attacks, and its practicality is evaluated on Ethereum-based networks.

Keywords: Blockchain, Web 3.0, Web 3.0 authentication, smart contract.

1 Introduction

The term *WEB 3.0* originally referred to the web characterized by ontology or semantic technologies [1, 2]. However, today, the definition of Web 3.0 has evolved and broadened to encompass blockchain-based internet environments [3]. This transition signifies a paradigm shift from the Web 2.0 era, which was dominated by centralized technologies, toward a decentralized landscape. Furthermore, the Web 3.0 environment is committed to delivering user-centric services underpinned by decentralized technologies [4], with a primary focus on safeguarding user data sovereignty [5].

In order to achieve user-centered services that align with the objectives of the Web 3.0 environment, it is crucial to bring about fundamental changes within the existing centralized systems [6–8]. One notable example is the development of decentralized data processing techniques, which involve the adoption of technologies such as distributed associative learning to replace conventional centralized data processing methods. In particular, there is an urgent need for research into authentication technologies that have traditionally relied on centralized servers of authorities, as they need to adapt to the decentralized nature of Web 3.0 environments. Authentication plays a pivotal role as a security measure, ensuring the protection of users in a landscape where all elements are distributed and stored.

Authentication serves as a fundamental cornerstone of online systems, functioning as the mechanism that establishes authorization for individuals or entities [9]. Authentication plays a pivotal role in enabling users to protect their data and assert their identity. In the context of Web 3.0 environments, characterized by the absence of centralized servers and authorities, the necessity for self-authentication becomes paramount. Furthermore, in the Web 3.0 environment, self-authentication empowers individuals to safeguard their data and rights, thereby laying the groundwork for the provision of user-centric services.

Existing centralized authentication technologies can be broadly categorized into three main groups: *proof by knowledge*, *proof by possession* and *proof by property* [10]. *Proof by knowledge* involves authentication through the use of identification passwords, while *proof by possession* relies on technologies like public key infrastructure (PKI) for authentication. On the other hand, *proof by property* encompasses authentication techniques based on biometric data, such as fingerprints.

Traditional centralized authentication techniques typically entail storing user information on centralized entities, such as servers or authorities. However, these techniques come with inherent limitations, including the risk of a user's credentials being exposed or leaked. Furthermore, they do not align with the user-centric services envisioned by Web 3.0, nor do they guarantee user data sovereignty.

In addition to traditional centralized authentication technologies, there is the emergence of decentralized identifiers (DID) technology that harnesses blockchain technology [11]. In DID technology, a DID document is generated and overseen by a DID controller, and user authentication is conducted through this DID document. DID technology represents a significant step towards decentralization when compared to traditional authentication techniques, as authentication information is verified using blockchain. However, it has a limitation in that it does not guarantee user data sovereignty, as the management of the DID document remains within the control of the DID controller.

Numerous authentication studies employing blockchain have been conducted [12–17]. While these studies have introduced various techniques for authenticating users using blockchain, they possess limitations that render them unsuitable for Web 3.0 environments.

This paper discusses the essential characteristics that should define Web 3.0 authentication technology and presents a Web 3.0 authentication based on these identified characteristics. The proposed approach comprises three key stages: *initialization*, serving as the initial authentication phase; *registration*, responsible for generating authentication tokens phase; and *authentication*, which encompasses the actual authentication procedure.

In this proposed approach, the absence of third-party intermediaries is a distinguishing feature, with only service users and service providers actively participating in the authentication process. Additionally, the system leverages blockchain-based components such as the *mother token*, *child token*, and *lock smart contract* to facilitate authentication.

The contributions of this paper can be summarized as follows:

- Redefining the characteristics of authentication technologies for the Web 3.0 environment, which has evolved from the Web 2.0 environment, was a pivotal step in our research.
- By delineating the key characteristics of Web 3.0 authentication technologies, we have not only provided valuable insights but have also paved the way for the development of a diverse range of new authentication techniques specifically tailored to the Web 3.0 environment.
- Our proposed approach, a decentralized and user-centric authentication technique, is designed to seamlessly align with the unique demands of the Web 3.0 environment.

The structure of this paper is as follows: Section 2 provides background information on Web 2.0 authentication and offers a definition of Web 3.0 authentication. Section 3 reviews the existing literature on authentication using blockchain technologies. Section 4 details the proposed Web 3.0

authentication approach. Section 5 delves into the experimental aspects of the research. Finally, Section 6 explains the conclusions from the findings and outlines potential avenues for future work.

2 Preliminaries

This section explores the attributes of conventional centralized authentication technologies while also establishing the criteria that should define Web 3.0 authentication.

2.1 Existing Centralized Authentication

In today's rapidly evolving landscape of internet technology, a myriad of authentication technologies have emerged [18]. This section introduces the three most prominent authentication techniques, namely: (1) password authentication, (2) PKI authentication, and (3) fingerprint authentication.

Password authentication serves as a quintessential example of proof by knowledge. It represents the most common and readily accessible form of authentication, widely adopted and tailored by service providers. The security of password authentication relies heavily on the complexity and length of the password itself [19, 20]. Service providers frequently encourage users to create passwords that include special characters and numeric sequences to enhance security.

PKI-based certificate technologies, exemplified by X.509, represent a classical proof by possession. These certificate-based authentication technologies require the participation of an authoritative entity, such as an administrator or a certificate authority (CA), who is responsible for generating and overseeing certificates. Moreover, these certificates must conform to specific formats, and the cryptographic protocols for communication need to be established in advance before certificate issuance and distribution [22].

Fingerprint authentication is one of the biometric authentication techniques, falling under the category of proof by property, which relies on biometric information such as a user's facial features, voice, or behavioral patterns to verify their identity [23]. Fingerprint authentication sets itself apart by mitigating the risk associated with losing an authentication certificate or entity, as it utilizes the user's biometric data. Furthermore, fingerprint authentication can be assessed using metrics such as false acceptance rate, false rejection rate, and equal error rate, as it operates within predefined optimal ranges and authentication thresholds [24].

2.2 Web 3.0 Authentication

This section explores the essential characteristics that authentication technology should possess within the context of the Web 3.0 environment, which represents a blockchain-based internet ecosystem. The primary objective of the Web 3.0 environment is to provide user-centric services through a variety of blockchain-based offerings, all while ensuring user data sovereignty. In other words, Web 3.0 authentication should guarantee user data sovereignty and exhibit properties suitable for a blockchain environment. To meet these requirements, Web 3.0 authentication should exhibit the following features: (1) *flexibility*, (2) *variability*, (3) *isolation*, and (4) *confidentiality*. Each of these features can be defined as follows.

Authentication flexibility. Authentication flexibility refers to the capability for users to authenticate themselves using information of their choosing. In existing authentication techniques, centralized institutions dictate the authentication information that users should provide. For instance, a centralized entity may require users to disclose their mobile phone numbers, email addresses, and more to access a service. In such situations, users are compelled to furnish their information whether they desire to or not. This can unintentionally lead to unwarranted encroachments on user privacy, which is incongruent with the fundamental goal of upholding user data sovereignty in the context of Web 3.0. To address this infringement on user data sovereignty arising from the mandatory submission of undesired authentication information, Web 3.0 authentication must inherently include the attribute of authentication flexibility. Through flexible authentication features, users can assert control over their data sovereignty by disclosing only the information they deem necessary to service providers.

Authentication variability. This concept implies that users should have the autonomy to create and delete authentication data or authentication entities as they see fit. Conventional credentials, managed by centralized authorities, leave users unaware of the fate of their data. Furthermore, since these authorities oversee user authentication data, they are not immune to various forms of breaches, exposures, and misuse [25]. Ultimately, this jeopardizes the integrity of user data sovereignty within the realm of centralized authorities. Web 3.0 authentication should incorporate authentication variability to enable the instantiation and removal of authentication information at the user's discretion, thereby reinforcing user data sovereignty.

Authentication isolation. Authentication isolation implies the need to safeguard a user's assets or data by preventing the multiple use of authentication information or authentication entities. In Web 3.0, a variety of decentralized services can be offered to users, leveraging the blockchain infrastructure. This environment provides the advantage of easy interconnection between services through the blockchain environment [26]. However, this convenience of interconnection also introduces security vulnerabilities that can potentially affect other services. For example, if the same private key used in a blockchain wallet is also used in a decentralized application, compromising the authentication entity associated with the private key could lead to the compromise of the user's virtual assets stored in the blockchain wallet and the user's data within the decentralized application. To mitigate such security risks, existing decentralized applications require intricate user authentication procedures [27]. To protect a user's assets or data, Web 3.0 authentication should encompass the characteristic of isolation of authentication, preventing the multiple use of authentication information or authentication entities.

Authentication confidentiality. Authentication confidentiality signifies that a user's authentication data should not be directly exposed to the distributed environment. Web 3.0 operates in a domain where blockchain-based services are prominent, meaning that all data is readily shared in a transparent manner due to the decentralized network. However, a user's authentication data should not be vulnerable to such unrestricted transparency, as it can be inherently connected to their personal information. Therefore, Web 3.0 authentication should incorporate the characteristic of authentication confidentiality, ensuring that a user's authentication data remains protected from direct exposure within the distributed environment.

3 Related Works

This section introduces existing research on blockchain-based authentication that will be reviewed and assessed for its alignment with the Web 3.0 authentication criteria outlined in the previous section. A summary of each research study is presented in Table 1.

Umoren's study [12] introduced an authentication technique that utilizes blockchain and fog computing. This approach employed smart contracts for both user registration and authentication procedures. Users' authentication data, including email, password, biometric data and Ethereum address, were securely hashed and stored on the blockchain. During authentication requests, a smart contract would validate the user's information stored on the blockchain, thereby confirming the user's identity.

Table 1 Comparison of related works

	[12]	[13]	[14]	[15]	[16]	[17]
Authentication flexibility	X	X	O	O	X	X
Authentication variability	X	X	O	O	O	O
Authentication isolation	X	O	O	O	X	O
Authentication confidentiality	O	X	O	O	X	O
Decentralization	O	X	X	X	X	O

One notable advantage of Umoren's approach is its alignment with the confidentiality feature of authentication, as it ensures that authentication data is not directly shared within the distributed environment. Additionally, it adheres to the decentralization aspect of authentication, as there is no involvement of centralized servers or authorities in the authentication process.

However, there are certain limitations to this proposed approach. Firstly, it does not fully meet the authentication flexibility as users are not given the option to select the specific information they wish to provide for authentication purposes. Secondly, it lacks the authentication variability since users are unable to dispose of their authentication information as needed. Lastly, the approach does not adhere to the isolation of authentication, as authentication information can be continually used once registered, potentially compromising user data security.

Muhammad Asif's blockchain-based authentication study [13] presented a novel approach to implementing blockchain-based authentication within the context of a smart city. In this study, a user initiates the authentication process with a smart contract, leading to the issuance of an access token. Subsequently, the user undergoes another authentication step with a key server, utilizing the access token received earlier. The key server continuously generates and provides access keys at specific intervals, allowing the user to access data upon request.

Asif's work successfully addresses the isolation aspect of authentication, mainly due to the dynamic nature of the keys used to access data, which change over time. However, there are notable limitations to this approach. Firstly, it falls short in terms of flexibility since users are required to provide specific information stipulated by the smart contract to obtain an access token. Secondly, it does not meet the authentication variability because users are unable to dispose of their authentication information as needed. Furthermore, it does not ensure the authentication confidentiality because users' authentication information is stored directly within the smart contract without encryption in Asif's proposed approach. Lastly, the paper does not adhere to the decentralization criterion of authentication due to the presence of a centralized authority referred to as the key server.

Yufan Wang conducted a study on user authenticating using blockchain and registration authority (RA) [14]. In this research, users have the capability to provide the RA with specific information they wish to use for authentication purposes. The RA, in turn, generates the user's key based on this information. Subsequently,the user stores the public key derived from the private key provided by the RA within a smart contract. When authentication is required, both the user and the service provider utilize the smart contract to generate a session key for mutual authentication.

Wang's work effectively addresses several aspects of authentication. It ensures authentication flexibility, as users can create a private key, the authentication entity, with their preferred information. Furthermore, it allows for the deletion and updating of the private key through the RA, thereby meeting the authentication variability. The authentication isolation is also maintained, as a new session key is generated for each authentication instance. Additionally, storing the user's public key within the blockchain, ensures that authentication information is not directly exposed.

Wang's research can be considered to satisfy all the characteristics of Web 3.0 authentication. However, it has a limitation in that it may not be entirely suitable for the Web 3.0 environment, which demands complete decentralization, as it involves a central authority, the RA, responsible for generating users' and service providers' private keys.

Xianbin Xu proposed an authentication scheme that combines fog computing and blockchain in a smart home environment [15]. This approach shares similarities with the study presented by Wang [14]. In this scheme, users provide their desired information to a trusted authority (TA) for authentication purposes. Subsequently, the TA generates a pseudo identity (PID) based on the information received from the user. User authentication is carried out through a combination of a blockchain smart contract and a fog node. The user initiates an authentication request to the blockchain smart contract, which then issues an access credential. Following this, the access credential undergoes authentication by the fog node, ultimately enabling the user to achieve authentication on the smart devices within the smart home.

Xu's paper effectively addresses several aspects of authentication. It ensures authentication flexibility, as users can create a PID, serving as an authentication entity, with their desired information. Moreover, it satisfies the authentication variability since a new PID can be generated through the

TA at any given time. By allowing the setting of validity times for access credentials, it also maintains the authentication isolation by preventing multiple uses of the authentication entity. Additionally, it ensures authentication confidentiality since the information used to generate the PID is not directly stored in the blockchain.

However, it can be argued that this approach may not be ideally suited for the Web 3.0 environment, which requires complete decentralization, due to the presence of the TA and fog node responsible for issuing PIDs and access credentials as authentication.

Alessio Catalfamo introduced a blockchain-based authentication scheme centered around the utilization of one-time passwords (OTPs) [16]. In this framework, the authentication microservice (AMS) is responsible for storing seed values in the blockchain, which are subsequently used by both users and service providers to create OTPs for authentication purposes. Users and service providers generate OTPs based on the seeds maintained by the AMS.

Catalfamo's research introduces an authentication technique that leverages OTPs, thereby effectively addressing certain aspects of authentication. Specifically, it ensures the authentication isolation by preventing the redundant use of authentication entities and satisfies the authentication variability due to the unique nature of OTPs which can be employed once and then discarded. Additionally, the approach maintains authentication confidentiality by encrypting and storing authentication information within the blockchain.

However, it's worth noting that this approach does not fully cater to the authentication flexibility, as OTPs are generated based on the seeds provided by the AMS, rather than allowing users to create OTPs using their preferred information. Moreover, it does not achieve the decentralization of the authentication process, as the AMS plays a central role in the generation of OTPs.

Mingli Zhang's research [17], similar to Alessio Catalfamo's study [16], presents an approach that utilizes blockchain-based OTPs. Zhang's study involves storing the user's public key and password information, which is generated using their private key, within a smart contract. When authentication is required, the user provides the OTP token, generated via the deployed smart contract, to the service provider. Subsequently, the service provider validates the OTP token via the smart contract.

In Zhang's study, it can be argued that the authentication variability is met since users have the ability to generate and discard OTP tokens-the authentication entity-according to their preferences. Furthermore, the authentication isolation is achieved due to the unique and newly generated nature of OTPs.

Zhang's approach also accomplishes the authentication confidentiality and the decentralization by ensuring that authentication information is not directly shared, and authentication relies solely on smart contracts.

However, it is important to note that this method does not fully address the authentication flexibility, as users are unable to directly choose the authentication information used in generating the OTP token, which serves as the authentication.

4 Proposed Scheme

This section introduces the proposed approach, *W3A*, a tokenized OTP authentication technique designed specifically for the Web 3.0 environment. *W3A* is meticulously designed to encompass all the essential characteristics of Web 3.0 authentication, as elucidated in Section 2.

4.1 Overview

As *W3A* is a decentralized authentication technique, the authentication process exclusively involves two entities: the service user and the service provider. Furthermore, the process relies on pivotal components, each elucidated below:

1. **Service user (su):** Serving as the central figure within *W3A*, the service user assumes the role of generating the mother token, the child token, and the lock smart contract. Crucially, the service user retains control over the child token, allowing for the inclusion of specific user information intended for the service provider.
2. **Service provider (sp):** The service provider functions as the entity responsible for delivering services to the service user. Upon receiving the child token from the service user, the service provider can access the user's designated information contained within the child token, facilitated through the utilization of the lock smart contract.
3. **Mother token (mt):** The mother token plays a pivotal role in managing and issuing child tokens. A one-to-many relationship exists between the service user and the mother token, while a similar one-to-many relationship is established between the mother token and child tokens.
4. **Child token (ct):** The child token is an authentication token that the service user provides to the service provider for authentication purposes. It maintains a connection with the mother token, with the service user

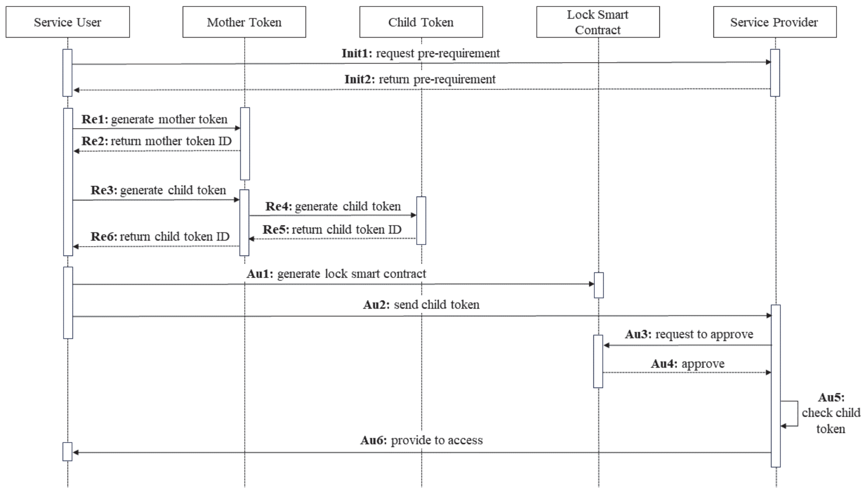
Service User
Mother Token
Child Token
Lock Smart Contract
Service Provider
Init1: request pre-requirement
Init2: return pre-requirement
Re1: generate mother token
Re2: return mother token ID
Re3: generate child token
Re4: generate child token
Re5: return child token ID
Re6: return child token ID
Au1: generate lock smart contract
Au2: send child token
Au3: request to approve
Au4: approve
Au5: check child token
Au6: provide to access

Figure 1 Sequence diagram of the entire W3A process.

having control over multiple child tokens. The child token structure comprises three key segments: header, body, and tail.

5. **Lock smart contract (lsc):** The lock smart contract stands as an integral component designed to safeguard the child token. It establishes direct links with the header, body, and tail segments of the child token. To access the body value of the child token, the service provider must successfully pass the lock smart contract verification. Additionally, the service user can only modify the header, body, and tail values following successful verification by the lock smart contract.

The proposed approach comprises three key phases: initialization, registration, and authentication. A concise overview of this process is presented in Figure 1. The initialization phase is the preparatory phase for the service user, where the service user generates the mother token, which plays a pivotal role in managing and issuing child tokens. The registration phase is where the service user generates a child token and a corresponding lock smart contract. These components are fundamental for the subsequent authentication process. The authentication phase marks the final phase in which the service provider validates the user's identity. This is achieved through interaction with the lock smart contract, facilitated by the child token. Detailed information on each of these processes is provided in the subsections. For clarity and reference, the notation used within each process is represented in Table 2.

Table 2 Notations table

Notation	Description
su	Service user
sp	Service provider
mt	Mother token
mt_id	Mother token ID
mt_sc	Mother token smart contract
ct	Child token
ct_id	Child token ID
lsc	Lock smart contract
$pr(x)$	Pre-requirement for access to SP service
h_ct	Header part of child token
b_ct	Body part of child token
t_ct	Tail part of child token
h_ctv	Header value of child token
b_ctv	Body value of child token
t_ctv	Tail value of child token
sv	Secret value
pr_an	Probability analysis
su_info	Service user information

4.2 Initialization Phase

This section introduces prerequisites set by the sp prior to embarking on the *W3A* process. In this phase, the sp formulates a set of conditions for utilizing a service, and disseminates it publicly to one or more su. By employing prerequisites, su and sp circumvent the necessity of exchanging data for authentication, provided they mutually agree on a standardized method of converting strings into numerical values. sp formulates and discloses a $pr(x)$ polynomial, permitting only those su capable of solving the $pr(x)$ polynomial to gain access to the service. Given that the $pr(x)$ generated by the sp is made accessible to the public, it must be structured in such a way that it does not inadvertently expose the personal information of the su.

When a sp receives a request from a su for the provision of a $pr(x)$ to access a service, it formulates and supplies a $pr(x) = \alpha + \sum_{k=1}^{\infty} n_k x^k$, where α represents a constant determined by the sp. Here, n signifies the total number of requirements established by the sp, while k represents the count of requirements generated by the sp. For instance, if the sp has previously committed to converting strings into numerical values using su and ASCII codes, and only extends access to the service to individuals who possess knowledge of the birth year of Bitcoin and the name of the head of the

Ethereum Foundation, a $pr(x)$ can be constructed in the following format: $Pr(X) = 10 + 2009x^2 + 1485x^2$.

4.3 Registration Phase

This section introduces how the su creates and utilizes mt, ct and lsc, which serve for authentication.

4.3.1 Mother token generation

The mother token, mt, is a token for the management purposes of the ct. The su can choose to create a new mt every time they use the services of different sp, or the su can use a single mt consistently for various sp services. The creation of the mt is facilitated through a blockchain smart contract, with a one-to-one correspondence between each mt and a smart contract.

In addition, it's important to note that the mt cannot be transferred or assigned to other users or even an sp. The unique identifier for an mt, mt_id, can be randomly generated based on smart contract settings or generated by the user through hash calculations using a seed value. Later, ct_id is generated through the mt_id, and ct_id can be proved to correspond to the generated ct from the mt.

4.3.2 Child token generation

The child token, ct, serves as the authentication certification provided directly by the su to the sp. For each authentication instance, the su generates a new ct that has the format $ct = (h\_ct, b\_ct, t\_ct)$.

Each ct possesses a unique identifier, ct_id, which is generated in the form of $ct\_id = Hash(mt\_id||Hash(h\_ctv||t\_ctv))$. t_ct represents the value corresponding to the tail component of the ct, created by su as $t_ctv = sv \oplus pr\_an$. b_ct signifies the value corresponding to the body part of the ct, where b_ct stores user information. su creates $b\_ctv = sv \oplus su\_info$ and stores it in b_ct. Access to b_ct requires verification by the lock smart contract (lsc). h_ct stands for the header of the ct, and su stores $h\_ctv = unixtime$ in h_ct. su can modify h_ctv, b_ctv, and t_ctv after lsc verification before passing ct to sp.

4.4 Authentication Phase

The authentication process of the su commences when the su initiates an lsc for ct authentication and transmits both the mt_id and ct to the sp.

4.4.1 Lock smart contract

The lsc plays a pivotal role in enabling the su to modify the ct value by validating the association between mt and ct. Additionally, lsc grants authorization to the sp for accessing the b_ctv. The algorithm of lsc is illustrated in Algorithm 1.

Algorithm 1 Lock smart contract process

```
Require: $mt\_id, ct, address$
  if $h\_ctv >$ current time then
      $ct\_value\_hash = Hash(h\_ctv, t\_ctv)$
      $ct\_id\_compare = Hash(mt\_id || ct\_value\_hash)$
      if $ct\_id == ct\_id\_compare$ then
          $mt\_owner = owner_check(mt\_id)$
          $ct\_owner = owner_check(ct)$
          if $mt\_owner == ct\_owner$ then
              provide write access to $h\_ctv, b\_ctv, t\_ctv$
          end if
          Provide read access to $b\_ctv$
          $t\_ctv$ to nil
      end if
  end if
```

The su and sp can operate the lsc using mt_id and ct. When the lsc receives an unlock request from both the su and sp, the lsc initially checks the h_ctv to determine if the ct is valid or not by comparing the current time stamp with h_ctv. For example, if the current time stamp exceeds h_ctv, it means the ct is not valid.

After that, the lsc verifies the relationship between mt_id and ct. The lsc generates $ct\_id' = HASH(mt\_id || HASH(h\_ctv || t\_ctv))$ using the mt_id provided by both the su and sp and checks if the generated ct_id' matches the ct_id of the received ct.

When the relationship between mt and ct is proved, lsc checks the owners of mt and ct. If the owners of mt and ct are the same, lsc determines that the su that created the ct is the same, and grants su the authority to modify h_ctv, b_ctv, and t_ctv. If su wants to revoke ct, su can change the h_ctv value to zero. If the h_ctv value is changed to zero, the remaining values b_ctv and t_ctv are changed to zero and cannot be used in the future.

lsc determines that the sp has been approached if the owner of mt and the owner of ct are different. lsc gives sp the right to read b_ctv if the valid time of ct remains. Thereafter, lsc prevents ct from being reused by changing the h_ctv value to zero and discarding it.

4.4.2 Authentication process

After the lsc is issued, su transmits mt_id and ct to sp. When sp receives ct, it first checks h_ctv. sp determines ct as valid if $h\_ctv > currenttime$. After that, sp obtains the b_ctv value by obtaining the right to read the b_ctv from lsc. sp approaches su_info using the t_ctv and the b_ctv.

In order to access su_info, sp obtains the sv value set by the user through $pr\_an \oplus t\_ctv$. sp can access su_info through $sv \oplus b\_ctv$ using the obtained sv. In this case sp cannot obtain accurate su_info unless su generates the t_ctv by inputting the correct pr_an, which means that user authentication has failed. When sp successfully obtains su_info, it looks at the corresponding content and completes the user authentication procedure.

5 Security Analysis and Performance Evaluation

This section analyzes *W3A* to assess its alignment with the characteristics of Web 3.0, as defined in Section 2. Additionally, this section includes a description of the security analysis and performance evaluation of the proposed approach.

5.1 W3A Analysis

5.1.1 Decentralized

Decentralization of authentication involves the elimination of entities like the trust third party (TTP) or certificate authority (CA) responsible for creating and managing authentication. It also creates an environment where a user's authentication information or credentials can be securely protected due to the absence of a single point of failure. If *W3A* were to adopt a centralized authentication approach, it would necessitate the presence of TTP or CA to manage the authentication. This centralized model would exist a single point of failure, potentially leading to the leakage of user information or credential.

As evident from the *W3A* approach, it involves only two entities, su and sp, and authentication occurs without the involvement of any other organizations. An su can independently create and manage the authentication entity, issuing an mt to establish the authentication entity and a ct for authentication purposes. Within the ct, the su can input desired su_info, and the lsc, responsible for managing the ct, is also issued by the su. Moreover, since *W3A* operates on a blockchain-based authentication approach, it is inherently resilient to single point of failure. Consequently, *W3A* can be characterized as a decentralized authentication approach, as it lacks institution such as TTP and CA, with authentication solely conducted by su and sp.

5.1.2 Authentication flexibility

Authentication flexibility refers to the user's ability to authenticate themselves using information of their own choosing. In cases where authentication flexibility is lacking, users may be compelled to provide information mandated by TTA, CA, or sp, even if it goes against their preferences. If *W3A* were to fall short in providing authentication flexibility, it would imply a process where users are required to furnish information demanded by a TTA, CA or SP, regardless of their personal preferences.

In *W3A*, sp requests su to provide pr_an, which is generated based on $pr(x)$. $pr(x)$ is a straightforward process aimed at verifying whether su meets the conditions for using the services offered by sp. It acts as a means to encrypt su's su_info without requiring predefined data exchange rules in a decentralized environment.

The structure of $pr(x)$ follows a simple polynomial form, such as $pr(x) = \alpha + \sum_{k=1}^{\infty} n_k x^k$. This essentially means that sp cannot demand complex or specific personal information from su. For example, if sp were to request su's age or phone number for various values of n, distinct pr_an values would be generated for each su. Consequently, sp cannot compel or request specific information from the user through $pr(x)$. Furthermore, it can be asserted that *W3A* adheres to the principle of authentication flexibility because su can include the su_info they wish provide in the ct and can modify su_info each time a new ct is generated.

5.1.3 Authentication variability

Authentication variability ensures that users have the freedom to create and dispose of their authentication information or credentials as they see fit. In the absence of authentication variability in *W3A*, there would be a requirement for the presence of a TTA or CA responsible for managing the authentication entity, or the authentication entity itself would need to be immutable.

As emphasized in Section 5.1.1, *W3A* operates independently, eliminating the need for a TTA or CA to manage the authentication entity. Moreover, users maintain full control over their issued ct through the lsc, enabling them to modify or delete their authentication entity information at their discretion. Thus, it is evident that *W3A* successfully satisfies the requirement for authentication variability.

5.1.4 Authentication isolation

Authentication isolation involves safeguarding a user's assets or data by preventing the multiple uses of authentication information or authentication entities. Failing to meet this requirement in *W3A* would mean that a single ct could be utilized by multiple sps.

However, *W3A* effectively addresses this concern by systematically discarding ct tokens through the lsc after each authentication process is completed. This design ensures that a single ct cannot be redundantly employed by multiple sps, preventing unauthorized access or the misuse of a user's credentials. In conclusion, *W3A* unequivocally adheres to the principle of authentication isolation.

5.1.5 Authentication confidentiality

Confidentiality of the certificate is crucial to protect certificate information from being directly shared in a distributed environment. If *W3A* were to fail in maintaining confidentiality, it would mean that su_info could potentially be accessed by unauthorized users beyond the su and sp involved in the authentication process.

To mitigate the risk of su_info exposure, *W3A* uses XOR operation. As explained in the approach, the certificate is structured as $ct = (h\_ct, b\_ct, t\_ct)$, where the user-configured sv is accessible via t_ctv, and su_info is retrievable from b_ctv through lsc. Additionally, accessing b_ctv requires authorization from lsc, which should provide proper ownership of the ct. In essence, *W3A* ensures authentication confidentiality by effectively preventing the direct sharing of su_info through robust encryption mechanisms.

5.2 Security Analysis

This section analyzes the security of *W3A*. It's important to note that this analysis does not cover scenarios directly targeting blockchain networks, such as 51% attacks, distributed denial of service attacks, and civil attacks. These type of attacks can vary significantly based on factors like the blockchain platform's performance, the consensus algorithm used, and whether the network is public or private. They are primarily influenced by blockchain network security measures and are not the main focus of this section.

Instead, this section will exclusively address scenarios in which an adversary ($\mathbb{A}$) attempts to compromise the security of *W3A*. This includes efforts to obtain or modify su_info through man-in-the-middle attacks, replay attacks, and brute-force attacks, as well as attempts to impersonate su to gain unauthorized access to sp.

The typical communication process in *W3A* can be summarized as follows:

- $req = su \rightarrow sp : pr(x)$:
  su requests $pr(x)$ from sp.
- $req = sp \rightarrow sup : pr(x)$:
  sp provides su with $pr(x)$.
- $mtg = su \rightarrow bc : mt$:
  su communicates with the blockchain to create mt.
- $ctg = su \rightarrow bc : ct$:
  su communicates with the blockchain to create a ct.
- $lscg = su \rightarrow bc : lsc$:
  su communicates with the blockchain to create an lsc.
- $au\_req = su \rightarrow sp : ct$:
  su delivers ct to sp for authentication
- $lscq = sp \rightarrow lsc : btcv$:
  sp authenticates ct to access $btcv$ via lsc.

5.2.1 Man-in-the-middle attack

A man-in-the-middle attack involves an $\mathbb{A}$ intercepting and potentially altering communication between two parties. For a man-in-the-middle attack to succeed, $\mathbb{A}$ must have the capability to modify the information contained in ct by intercepting it during communication. During the req and res processes, $\mathbb{A}$ could attempt to alter $pr(x)$ through such a man-in-the-middle attack.

However, it's crucial to emphasize that $pr(x)$ is publicly available information, accessible to both the su and the sp. Any attempts by $\mathbb{A}$ to tamper with $pr(x)$ could raise suspicions since the altered $pr(x)$ would not match the version initially provided by sp. Moreover, as the modified $\mathbb{A}_{pr(x)}$ by $\mathbb{A}$ inherently differs from the original $sp_{pr(x)}$ supplied by sp, even if the $pr(x)$ value is changed, it does not impact the authentication process's outcome.

Other communication processes, such as $mtg, ctg, lscg, au\_req$, and $lscq$, rely on blockchain-based communication, which incorporates private key-based signature technology. For instance, during the au_req communication, the su engages by transmitting a signature in the form of $sig(au\_req, pk)$ that has been signed with their unique private key to the sp. In essence, for an $\mathbb{A}$ to interfere with the au_req process, they would need to gain control of su's private key, a task that presents a substantial barrier and renders a successful man-in-the-middle attack unfeasible.

5.2.2 Replay attack

A replay attack is a type of attack where an $\mathbb{A}$ intercepts and attempts to reuse valid authentication information or messages exchanged in the communication process. To execute a successful replay attack, $\mathbb{A}$ must gain control of the ct. However, as discussed in the section on man-in-the-middle attacks, $\mathbb{A}$'s ability to access user information by seizing ct is restricted. Even if $\mathbb{A}$ manages to obtain the data within ct, it's important to note that *W3A* adheres to the principles of authentication variability and isolation.

In *W3A*, the su possesses the capability to modify the contents of ct at their discretion. This means that if su alters the information contained in ct, any previously acquired data by $\mathbb{A}$ becomes obsolete, rendering it ineffective for authentication purposes. Furthermore, the replay attack by $\mathbb{A}$ is thwarted due to ct having the attribute of a one-time password.

Therefore, the attribute of variability and isolation, and the one-time password attribute of ct collectively prevent the success of a replay attack by $\mathbb{A}$.

5.2.3 Brute force attack

The $\mathbb{A}$ may attempt to access the b_ctv by launching a brute-force attack to obtain the pr_an. If $\mathbb{A}$ successfully acquires the pr_an, they can access the sv by the su through the operation of $t\_ctv \oplus pr_an$. $\mathbb{A}$ may also try to deceive the lsc by providing the connection between mt_id and ct_id, both of which are publicly available on the blockchain network. This attempt is made in an effort to again access to su_info via the operation of $b\_ctv \oplus sv$.

However, it's essential to note that this attack cannot succeed due to the rigorous security measures in place within *W3A*. The lsc not only verifies the connectivity between mt_id and ct_id but also validates the ownership of the ct. Ownership of mt and ct held by su is guaranteed by the consensus of all participants in the blockchain network. This means that $\mathbb{A}$ cannot deceive the lsc into believing they own mt or ct unless they gain control of over 51% of the blockchain network, an extremely challenging and unlikely scenario.

5.3 Performance Evaluation

This section presents the performance evaluation and analysis based on the results obtained from applying the proposed approach to different blockchain networks within the Ethereum family. The performance evaluation was conducted by measuring two key time intervals: the time taken from mt generation to ct generation during the *registration phase* and the time taken from

lsc generation to ct delivery for sp authentication during the *authentication phase*.

For the performance evaluation, three distinct blockchain networks were used: the Ganache network [28], the Hyperledger Besu network [29], and the Goerli network [30]. Here's detailed breakdown of the experiment setup for each network:

1. Ganache network
    - Ganache serves as an Ethereum local test network
    - The experiment involved a fixed number of eight nodes
    - Block generation was configured to occur every one second.
2. Hyperledger Besu network
    - The Hyperledger Besu network is a private blockchain network based on the Ethereum platform.
    - Similar to the Ganache network, this experiment also utilized eight physical nodes
    - The istanbul BFT (IBFT) consensus algorithm was used, with block generation scheduled every one second.
3. Goerli network
    - The Goerli network is an Ethereum public test network closely resembling the Ethereum mainnet in terms of its settings
    - It implements the proof of stake (PoS) consensus algorithm.

The goal was to assess the performance of the proposed approach across networks with varying characteristics. By conducting experiments on both private (Ganache and Besu) and public (Goerli) networks, the experiment aimed to gain insights into how the system's performance in influenced by different network configurations and consensus algorithms. Each experiment was repeated 100 times, and the experiment measured the average time and standard deviation of the elapsed time in each case.

The results of measuring the registration phase time are depicted in Figure 2, where the x-axis represents each network, and the y-axis represents the corresponding elapsed time. In the Ganache network, the average registration phase time is 2.20 seconds, with a standard deviation of 0.17 seconds. For the Besu network, the average time is 2.12 seconds, with a standard deviation of 0.30 seconds. In contrast, the Goerli network, resembling a public testnet, exhibits significantly different performance, with an average registration phase time of 36.22 seconds and a standard deviation of 15.35 seconds.

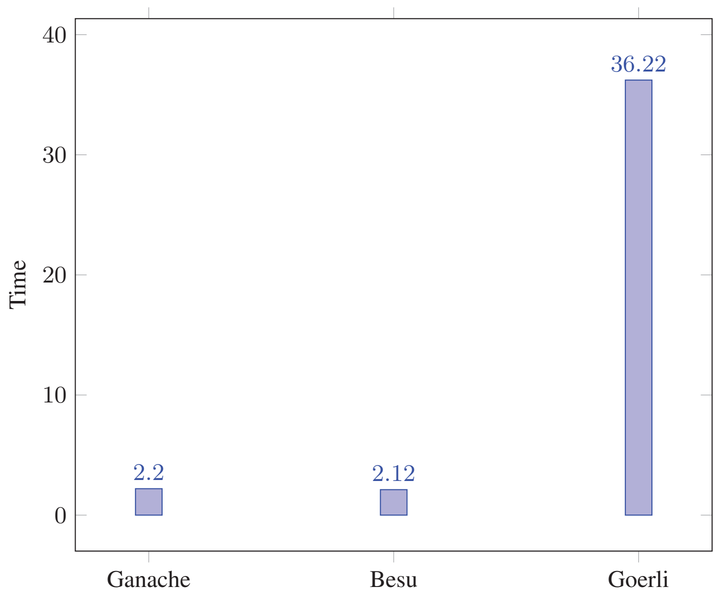

Figure 2 Elapsed time of registration phase.

These results indicate that there is no notable performance difference between the Ganache and Besu networks, both of which are virtual and private network environments. However, in the Goerli network, which is a testnet of Ethereum and resembles the Ethereum public network, the elapsed time difference compared to Besu is approximately 17 times, with a notable higher standard deviation than observed in other networks.

The results of measuring the elapsed time taken for the authentication phase are presented in Figure 3. Similar to Figure 2, the x-axis represents each network, and the y-axis represent elapsed time. In the authentication phase, the Ganache network exhibited an average time of 4.39 seconds with a standard deviation of 0.13 seconds. For the Besu network, the average time was 4.22 seconds with a standard deviation of 0.33 seconds, while the Goerli network had an average time of 67.42 seconds with a standard deviation of 23.11 seconds.

Like the registration phase, there is no significant performance difference between Ganache and Besu. However, Goerli was found to be about 15 times slower than Besu, with the highest standard deviation.

After evaluating the performance of both the registration and authentication phases, it is evident that the proposed approach is significantly

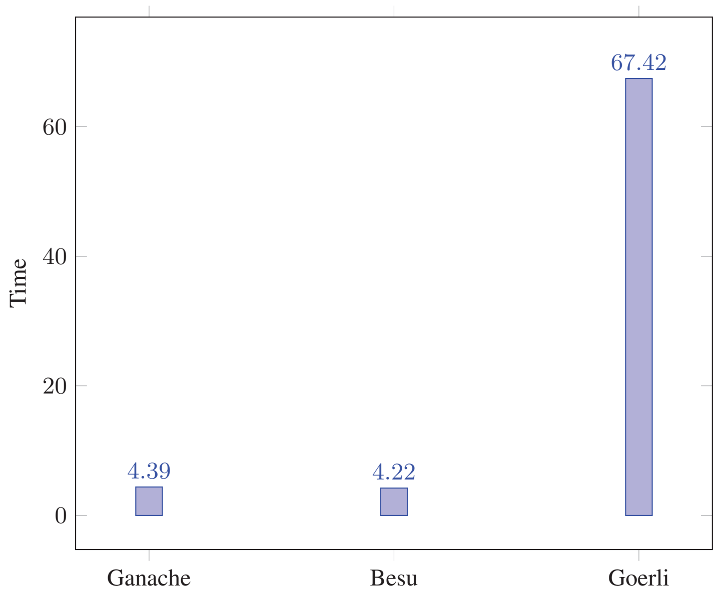

Figure 3 Elapsed time of authentication phase.

influenced by the performance and size of the underlying blockchain network. This influence arises due to the reliance on the blockchain network for all processes without external server or organization intervention. Notably, the average time and standard deviation tend to increase with the number of nodes and the network's complexity. In summary, a stable blockchain infrastructure is crucial for the seamless application of the proposed approach to various services.

5.4 Conclusion

With the emergence of Web 3.0, characterized by a user-centric internet environment, the demand for authentication techniques suited to this new era is on the rise. This paper addresses this need by defining the essential features that Web 3.0 authentication should embody and presenting a Web 3.0 authentication techniques designed to align with these features.

The proposed approach leverages smart contracts to generate mother tokens and child tokens, facilitating authentication through a lock smart contract. The approach successfully fulfills all the Web 3.0 authentication criteria outlined in this paper and demonstrates robust security against potential threats like man-in-the-middle, replay, and brute-force attacks. Furthermore, the performance experiments were conducted on Ethereum-based networks to evaluate the practicality of the proposed approach.

The Web 3.0 authentication features and techniques introduced in this paper are poised to serve as a foundational framework for future developments in Web 3.0-based authentication technologies. Subsequent research endeavors will focus on enhancing the blockchain infrastructure to further optimize the speed and efficiency of the proposed approach.

Acknowledgment

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2024-RS-2023-00259099) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation). This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2024-00397538, Development of public opinion polling technology based on web3 that ensures fairness, anonymity, and transparency, 50%).

References

[1] O. Lassila, and J. Hendler. Embracing Web 3.0. *IEEE Internet Computing*, 11(3):90–93, 2007.

[2] J. Hendler. Web 3.0 Emerging. *Computer*, 42(1):111–113, 2009.

[3] C. Chen, L. Zhang, Y. Li, T. Liao, S. Zhao, Z. Zheng, H. Huang, and J. Wu. When Digital Economy Meets Web3.0: Applications and Challenges. *IEEE Open Journal of the Computer Society*, 3:233–245, 2022.

[4] S. Yangm, and M. Li. Web3.0 Data Infrastructure: Challenges and Opportunities. *IEEE Network*, 37(1):4–5, 2023.

[5] A. Suryavanshi, A. G, M. Babu T. N, R. M, and A. Haq N. The integration of Blockchain and AI for Web 3.0: A security Perspective. *2023 4th International Conference on Innovative Trends in Information Technology (ICITIIT)*, 2023.

[6] C. Guan, D. Ding, and J. Guo. Web3.0: A Review And Research Agenda. *2022 RIVF International Conference on Computing and Communication Technologies (RIVF)*, 2022.

[7] Z. Liu, Y. Xiang, j. Shi, P. Gao, H. Wang, X. Xiao, B. Wen, Q. Li, and Y. Hu. Make Web3.0 Connected. *IEEE Transactions on Dependable and Secure Computing*, 19(5):2965–2981, 2022.

[8] P. S. S, and A. Kumar. Web3.0 E-Commerce Decentralized Application. *SSRN*. Available online: https://ssrn.com/abstract=4268681.

[9] S. Y. Lim, P. T.Fosting, A. Almasri, O. Musa, M. L. M. Kiah, T. F. Ang, and R. Ismail. Blockchain technology the identity management and authentication service disruptor: A survey. *International Journal on Advanced Science Engineering and Information Technology*, 8:1735–1745, 2018.

[10] T. Y. C. Woo, and S. S. Lam. Authentication for distributed systems. *Computer*, 25(1):39–52, 1992.

[11] Decentralized Identifiers (DIDs) v1.0. *W3C*. Available online: https://www.w3.org/TR/did-core/.

[12] O. Umoren, R. Singh, Z. Pervez, and K. Dahal. Securing Fog Computing with a Decentralised User Authentication Approach Based on Blockchain. *Sensors*, 22(10):3956, 2022.

[13] M. Asif, Z. Aziz, M. B. Ahmad, A. Khalid, H. A. Waris, and A. Gilani. Blockchain-Based Authentication and Trust Management Mechanism for Smart Cities. *Sensors*, 22(7):2604, 2022.

[14] Y. Wang, X. Jia, Y. Xia, M. K. Khan, and D. He. A blockchain-based conditional privacy-preserving authentication scheme for edge computing services. *Journal of Information Security and Applications*, 70:103334, 2022.

[15] X. Xu, Y. Guo, and Y. Guo. Fog-enabled private blockchain-based identity authentication scheme for smart home. *Computer Communications*, 205:58–68, 2023.

[16] A. Catalfamo, A. Ruggeri, A. Celesti, M. Fazio, and M. Villari. A Microservices and Blockchain Based One Time Password (MBB-OTP) Protocol for Security-Enhanced Authentication. *2021 IEEE Symposium on Computers and Communications (ISCC)*, 2021.

[17] M. Zhang, L. Wang, and J. Yang. A Blockchain-Based Authentication Method with One-Time Password. *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, 2019.

[18] S. W. Shah, and S. S. Kanhere. Recent Trends in User Authentication - A Survey. *IEEE Access*, 7:112505–112519, 2019.

[19] W. Ma, J. Campbell, D. Tran, and D. Kleeman. Password Entropy and Password Quality. *2010 Fourth International Conference on Network and System Security*. 2010.

[20] H. Murray, and D. Malone. Evaluating password advice. *2017 28th Irish Signals and Systems Conference (ISSC)*, 2017.

[21] M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. *rfc2560*. Available online: https://www.rfc-editor.org/rfc/rfc256 0.

[22] C. Mitchell. PKI standards. *Information Security Technical Report*, 5(4):17–32, 2000.

[23] Z. Rui, and Z. Yan. A Survey on Biometric Authentication: Toward Secure and Privacy-Preserving Identification. *IEEE Access*, 7:5994–6009, 2018.

[24] D. Bhattacharyya, R. Ranjan, F. Alisherov A, and M. Choi. Biometric Authentication: A Review. *International Journal of u-and-e-Service, Science and Technology*, 2(3):13–28, 2009.

[25] S. Cucko, and M. Turkanovic. Decentralized and Self-Sovereign Identity: Systematic Mapping Study. *IEEE Access*, 9:139009–139027, 2021.

[26] C. Antal, T. Cioara, I. Anghel, M. Antal, and I. Salomie. Distributed Ledger Technology Review and Decentralized Applications Development Guidelines. *Future Internet*, 13(3):62, 2021.

[27] 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*, 6:53019–53033, 2018.

[28] Ganache, One Click Blockchain. *Truffle Suite*. Available Online: https://trufflesuite.com/ganache/.

[29] Hyperledger Besu Ethereum client. *Hyperledger Besu*. Available Online: https://besu.hyperledger.org/

[30] Goerli Testnet. Available Online: https://goerli.net/

Biography

Jungwon Seo holds a Ph.D. in Software Engineering & Blockchain from Sogang University. Additionally, he earned a Master's degree in Computer Science & Engineering from Sogang University in March 2020, specializing in Software Engineering & Blockchain. He also graduated with a Bachelor's degree in Management Information Systems from the State University of New York at Buffalo's Business Department in May 2016.