applied sciences

MDPI

Article

A Secure and Decentralized Authentication Mechanism Based on Web 3.0 and Ethereum Blockchain Technology

Adrian Petcu *, Bogdan Pahontu, Madalin Frunzete and Dan Alexandru Stoichescu

Faculty of Electronics, Telecommunications and Information Technology, University Politehnica of Bucharest, 061071 Bucharest, Romania

* Correspondence: adrian.petcu@stud.etti.upb.ro

Abstract: Over the past decade, there has been significant evolution in the security field, specifically in the authentication and authorization part. The standard authentication protocol nowadays is OAuth 2.0-based authentication. This method relies on a third-party authentication service provider with complete control over the users' data, which it can filter or modify at will. Blockchain and decentralization have generated much interest in recent years, and the decentralized web is considered the next significant improvement in the world wide web (also known as Web 3.0). Web3 authentication, also known as decentralized authentication, allows for the secure and decentralized authentication of users on the web. The use cases for this technology include online marketplaces, social media platforms, and other online communities that require user authentication. The advantages of Web3 authentication include increased security and privacy for users and the ability for users to have more control over their data. The proposed system implementation uses Ethereum as the blockchain and a modern web stack to enhance user interaction and usability. The solution brings benefits both to the private and the public sector, proving that it has the capability of becoming the preferred authentication mechanism for any decentralized web application.

Keywords: web3; authentication; authorization; access control; trust

Citation: Petcu, A.; Pahontu, B.; Frunzete, M.; Stoichescu, D.A. A Secure and Decentralized Authentication Mechanism Based on Web 3.0 and Ethereum Blockchain Technology. *Appl. Sci.* **2023**, *13*, 2231. https://doi.org/10.3390/app13042231

Academic Editor: Paolo Renna

Received: 15 December 2022
Revised: 4 February 2023
Accepted: 7 February 2023
Published: 9 February 2023

1. Introduction

To properly understand what Web3 authentication means, we will start the following article by diving into traditional login methods used in Web2, where the application/website interaction with the user consists of either validating user credentials or validating a third-party key generated by an external system or application.

Over the decades, we have seen an evolution of web-based applications from the initial stages at 1.0, where websites would serve static content with which the users could interact statically (the 1900s–2000s). Web 2.0 brought an advantage to the field, allowing users to interact with their content and dynamically change their surfing experience of an application or website (the 2000s–2020s). Service providers and application owners would centralize the user data and provide logistic support and allow for increased user interactivity. Web3 (the 2020s–present) promises to bring the playing field to a new level with decentralized service providers and an anonymous identity for all users.

User identities, consumer habits, navigation history, and patterns in behavior are generally stored and can be used to predict behaviors or to suggest possible areas of interest in fulfilling the user's desires.

With more and more websites collecting user and behavioral data without the users' consent or knowledge, a problem emerged such that users were no longer in control of their data and information. Instead, the European Union created a regulation specifically for this purpose in 2014 called GDPR (General Data Protection Regulation).

This paper will explore a novel way of authenticating with a focus on decentralization by using the tools and concepts already used in the Ethereum blockchain ecosystem, bridging the gap between on-chain and off-chain resources . The first section will discuss current

secure authentication methods, Web 3.0 definitions, history, and practical applications. The second section goes through our proposed technical stack for obtaining the desired authentication method. Afterward, we explain the advantages and disadvantages of Web 2.0 authentication versus Web 3.0.

Finally, we present a practical implementation of the proposed solution along with a step-by-step guide as well as the benefits and challenges of using an anonymous login system with no data that can be traced back to the user. A study was carried out to compare login times of the proposed implementation against similar login mechanisms.

Since the proposed login mechanism is one-factor, we observe the times required to perform a successful login using other one-factor relevant systems such as SMS login and also compare time differences between mobile phone wallet solutions and browser wallet software wallet solutions.

2. Related Work and Secure Authentication Mechanisms

The most popular authentication mechanism relies on the user entering a username/e-mail and password combination in order to gain access to the system. However, if the password becomes compromised, the user might lose access to the system forever. To enhance authentication security and increase account recoverability, many platforms resort to two-factor authentication (2FA) which is a security measure that aims to provide an additional layer of protection for online accounts and services. The concept of 2FA involves implementing a multi-factor authentication process, which requires the user to provide two distinct forms of identification. This approach is intended to mitigate the risk of unauthorized access to an account, as it makes it more difficult for an attacker to bypass the authentication process. There are several different types of 2FA mechanisms, of which we can mention the following:

SMS-based 2FA: This type of 2FA involves sending a one-time code via text to a users' phone number. The user then enters this code to access their account. This method is simple to use but can be vulnerable to SIM-swapping attacks, where an attacker can hijack the users' phone number and intercept the code.

A 4-digit code is generated on the server and stored along with the generation date. The code is then sent to the users phone via SMS, and upon entering it into the desired platform, successful authentication is achieved.

Time-based one-time passwords (TOTPs): This type of 2FA uses a software application, such as Google Authenticator or Authy, to generate a new code every 30 s. The user enters this code to access their account. This method is more secure than SMS-based 2FA because the code is generated on the users' devices and is not transmitted over a network.

The users need to have a pre-installed application used to generate codes that follow a pattern. The pattern is synchronized with the server pattern by scanning a QR code generated by the backend server. After logging in with the predefined credentials (username and password), a prompt needs to be filled in with the code generated on the application.

Biometric-based 2FA: This type of 2FA uses the users' unique physical or behavioral characteristics, such as fingerprint or facial recognition, to verify their identity.

Security Key: This type of 2FA uses a physical device, such as a USB key, that the user must plug into their computer or insert into their phone to access their account.

Using two-factor authentication (2FA) among websites and online services is increasingly relevant in cyber security. The frequency of 2FA implementation can vary depending on various factors, including the nature of the information being accessed and the industry in which the website or service operates. For example, a study conducted by Google in 2017 found that 2FA adoption rates had increased over time, with an adoption rate of 15% among all Google accounts.

Additionally, a study by Duo Security in 2019 revealed that while a majority (70%) of the top 100 most-visited websites in the United States offered 2FA, only a tiny percentage (3%) of users had enabled it on their accounts. This highlights the need for further education

and awareness surrounding the importance and benefits of 2FA in protecting personal and sensitive information.

There is a trend in dropping the username and password combination and just using a single-factor authentication method to decrease the time required to create an account on a specific platform. Instead of entering the registration information (i.e., email and password) when using the application for the first time, the users are prompted to enter their phone number. Validation of the phone number is achieved by entering the SMS code received. As previously stated, to achieve a faster onboarding and login time, multiple applications or websites have started using authentication based only on the SMS code, without needing a username or password. However, recovering the account details might prove difficult if no email is associated with the account.

Our study aims to explore the feasibility of authentication systems based on complete anonymity, thus bridging the gap between operations performed on and off the blockchain.

3. Web 3.0

3.1. What Is Web 3.0?

Web3, also known as Web 3.0, is an idea of the next World Wide Web version, which focuses on decentralizing the data and a token-based economy [1]. Decentralization empowers peer-to-peer information exchange, eliminating intermediaries and removing third-party entities that might control the data. In public blockchains, cryptocurrencies play a significant role in the general scheme, as the token economy facilitates the centralization model. Information is stored in a distributed ledger outside the authority of a single point of failure or entity. Smart contracts allow for immutable code, allowing transparency and traceability.

Since Web 3.0 [2] proposes a decentralized architecture that is open to everyone, its connection with blockchain is obvious. Public blockchains operate independently, and data are never controlled by one single entity. Leveraging the power of blockchain, we can enhance Web 2.0 toward reaching full decentralized information and Web 3.0.

3.2. Web 3.0 Layers

Web 3.0 architecture is slightly different than traditional architectures. It has a high focus on decentralization and is comprised of the following layers:

- Application layer: Users interact with decentralized applications (dApps) built on the blockchain. dApps can be used for various purposes, such as online marketplaces and social media platforms.
- Presentation layer: Tools and libraries used to interact with the blockchain.
- Blockchain interaction layer: Application interfaces (APIs) and graphical interfaces used for debugging the blockchain's current state.
- Network layer: Ensures communication between nodes.

As also mentioned by [3] in Figure 1, we can see the multiple layers that compose blockchain-based decentralized applications. Our focus will be on the first layer, connecting the front-end layer to the back-end, leveraging the signature from the user-installed software or hardware wallet.

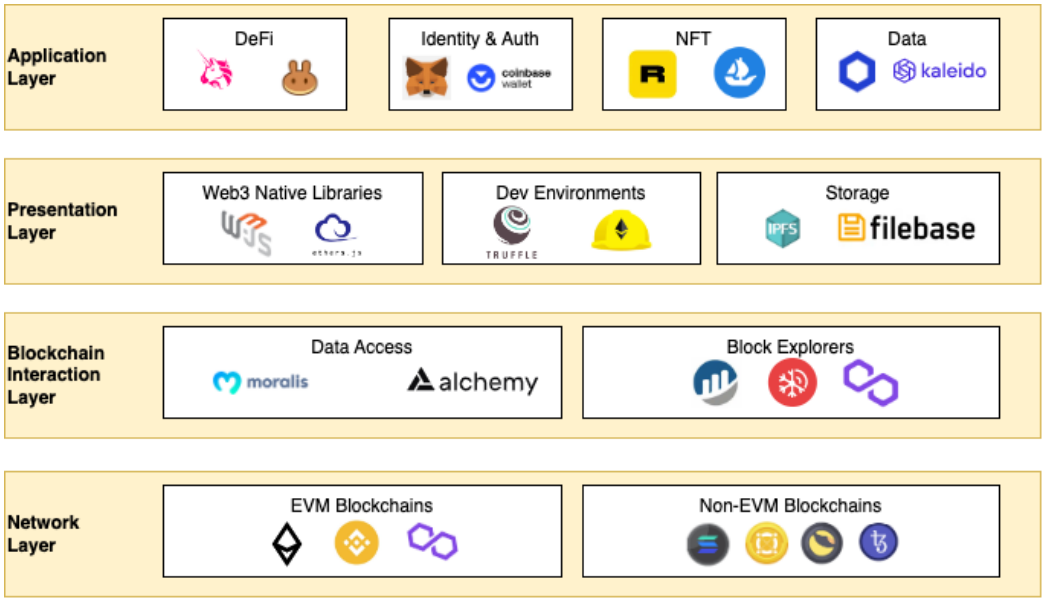


Figure 1. Blockchain application layers.

3.3. Web 3.0 Applications

3.3.1. DeFi: Decentralized Finance

Decentralized finance, in short DeFi, aims to revolutionize the way financial systems work, removing intermediaries and empowering individuals and peer-to-peer transactions at reduced costs in a matter of seconds.

DeFi heavily relies on tokens [4] and smart contracts empowered by blockchain as opposed to traditional finance systems. Access to finances in a DeFi environment is primarily anonymous and can be done in a matter of seconds without a central authority to validate users' identities.

Some centralized finance concepts have also been implemented in the DeFi environment. However, since the users' identity is unknown, concepts such as collateral and interest have been adjusted to ensure that the systems work with zero knowledge of user-related information.

Lending and borrowing tokens require token liquidity [5], whereas, in real life, a physical asset would be required.

Decentralized exchanges (DeX) have paved the way for an innovative way of trading tokens. In centralized exchanges, authorities would facilitate trading between peers, and market makers would stimulate trading whenever a buy order was not present to fulfill the order [6]. On the other hand, matching buy-sell orders in a decentralized environment is relatively tricky, so the entire system relies on liquidity pools. If trading X token for Y token, there should be enough Y tokens available in the liquidity pool to fulfill the order. The balance between the two pools would eventually determine the buying price.

3.3.2. NFTs: Non-Fungible Tokens

Non-fungible tokens (NFTs) are smart contracts that take the form of digital assets that leave a fingerprint on the blockchain.

Fungibility is the ability of goods and assets to be interchangeable. For example 1 dollar = 1 dollar. Non-fungibility means that all assets are unique, similar to paintings.

Non-fungible assets have multiple applications [7] and even though there has been a highly speculative bubble for digital art, their use can allow the asset holder access to certain events or resources. Each owner's identity is stored in the blockchain and can be verified via smart contracts.

There have been multiple marketplaces that have emerged that would allow the exchange of such assets.

Some practical uses of NFTs are the following:

- Digital concert tickets.
- Proof of ownership.
- Real estate.
- Intellectual property and patents.

3.3.3. DAOs: Decentralized Autonomous Organization

The concept of decentralized autonomous organizations (DAOs) [8] is relatively simple. A democratic structure is formed, and voters would be all users holding crypto tokens.

There is no hierarchy, leadership, or boards, and since blockchain allows transparency and immutability, this concept provides increased trust.

DAOs heavily rely on smart contracts to ensure transparency and anonymity as opposed to traditional structures, such as company boards of executives. As a result, the decision power is spread to the community.

4. Technical Stack

4.1. Ethereum-Based Wallets

Ethereum wallets are applications that allow interaction with Ethereum [4] accounts. For example, an Ethereum account has access to the private key used to perform operations on behalf of the public key, which is the public address to which funds and assets can be transferred.

Wallets can be either software or hardware. For example, hardware wallets use an external device, such as a USB drive, to store users' private keys, whereas software wallets store the keys inside the computer/phone memory.

Most software wallets are built as browser add-ons, which makes it easier for decentralized apps to interact with them. However, some software wallets are in the form of phone applications. For example, the interaction with wallets in the form of phone applications can be tedious and may require external services to facilitate communication. The most popular third-party providers for mobile phone wallet connectivity are WalletConnect and WalletLink.

An example of interaction with mobile phone wallets can be seen in Figure 2. The GUI (graphical user interface) displays a QR code which, upon scanning, triggers a connection request. After the connection is successful, it is stored on a third-party server which facilitates communication between the dApp and the wallet, similar to browser extension wallets.

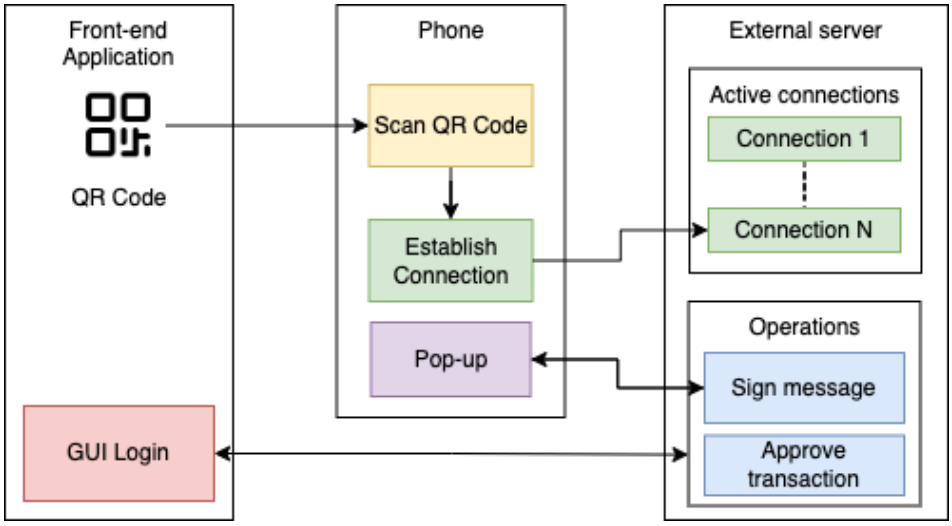


Figure 2. Mobile phone wallet interaction flow.

WalletConnect relies on websockets to relay messages from and to the mobile phone wallet and the decentralized application. When the QR code from Figure 2 is scanned on the mobile device, a connection is established and stored on the relay server (external server). According to the specifications of WalletConnect, "The Bridge Server acts as pub/sub controller which guarantees that published messages are always received by their subscribers". This means that we can assume that communication is efficient and uninterrupted.

Ethereum addresses are unique, case-insensitive strings of 42 hexadecimal characters that are produced from the private key. They represent an account on the blockchain of Ethereum. The generation of an Ethereum wallet address is done by using the private key (64 hexadecimal characters). Once generated, the private key must be stored securely, either by using an old fashioned pen and paper or by using specialized wallet management systems (hardware or software). These systems usually enhance the user experience by displaying the current assets and their balance, wallet address and facilitate communication with decentralized applications.

In order to generate an Ethereum wallet address, a random private key [9] (64 characters) is generated. An 128 (hex) character is derived from this private key. Upon applying a kekkak256 algorithm to the private key, we obtain a 64 character (hex) string and by using the last 40 characters of this string and prefixing them with 0x, the final wallet address is obtained. The wallet address and/or private key are not hardware related and can be generated from any machine or operating system.

4.2. Metamask

Metamask [10] is a popular blockchain software wallet that connects to Ethereum-based blockchains and exposes a javascript API that can interact with the blockchain via a user signature.

As seen in Figure 3, decentralized application developers leverage the interfaces exposed by software wallets to develop applications that may communicate directly to the blockchain without needing a back-end to serve requests. For example, smart contract function calls can be executed via Web3 connections, and transactions can be approved via the user wallets.

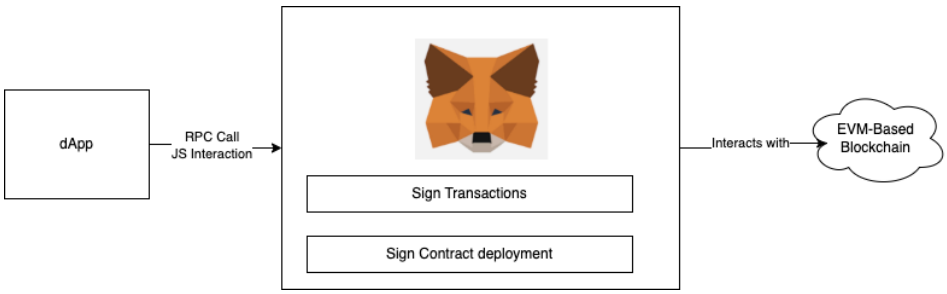


Figure 3. Metamask interaction.

Users can create wallets or import existing ones using Metamask, then they can start to approve blockchain transactions [11], which would either be transfers or smart contract interactions.

Metamask is considered safe, as it is open source, and storing the users' PVKI (pin verification key indicator) is done on the local machine.

Proof of owning the digital identity can be done either on-chain or off-chain through signing nonces (number used only once) via PVKI via ECDSA [12] (elliptic curve digital signature algorithm). Services can verify this signature by decrypting the message and validating its contents.

4.3. JSON Web Token (JWT)

The JSON web token is the most popular standard for validating users' identities without revealing personal or sensitive information in the exchange.

Conventionally, sessions were used to store the authentication state of users. For example, sessions would be stored server-side and accessed via cookies in the browser.

Since modern architectures rely on data and information decoupling, using sessions creates a single point of failure on the memory storage layer.

JWTs are generated after the user identity has been validated by the server, and usually contain information such as username, claims, and expiration time. JWT [13] signatures cannot be manipulated since they are generated by encrypting the data with a specific key. Thus, any alteration or manipulation would render the key invalid.

JWTs can be validated by multiple systems and are usually short-lived to avoid potential hacker access to the key, thus gaining access to the users' systems.

4.4. Two-Factor Authentication

Multi-factor authentication is a term used to describe the need for a user to provide a second layer of credentials to validate that he is the actual owner of an account. This is usually done via SMS messages or by entering secret codes that can be generated exclusively by a user-held device or software.

5. Authentication in Web 2.0

There are multiple ways to authenticate in any web-based or user-installed applications, either based on user credentials or certificates or relying on second-factor validation of users' identities.

- **Username and password** authentication relies on users knowing a combination of public usernames and secret passwords to gain access to systems.
- **Multi-factor authentication** adds an extra layer of security on top of username–password authentication using a physical device or a software token to generate a code that validates that the user is the owner of the account.
- **Certificate-based authentication** relies on certificates installed on the users' machine to gain access to a protected system.
- **Third-party login** can replace the username-password combination completely by trusting a 3rd party provider to manage users' identities.

All of the authentication methods in Web 2.0 have a centralized identity management system that stores user credentials and/or private information owned by the user.

6. Authentication with Web 3.0

Since most of the decentralized applications rely on user anonymity, systems must rely on only the proof-of-ownership [14] of the wallet address to unlock access to certain features or pieces of functionality.

Web3 [15] authentication is the starting point for most decentralized applications, and it leverages the built-in mechanisms in hardware or software wallets, such as signing transactions and messages. Our study relies on the message-signing capability to verify that the user has access to the private key.

6.1. Use-Cases

Web3 authentication allows bridging web2 systems to web3. However, some scenarios exist where, only using the blockchain wallet address, websites can act as intermediaries between different transactions. However, these transactions eventually end up being performed on-chain, and having a visual representation of an action is desired [16].

Such examples include the following:

1. **NFT marketplaces** rely on websites that bring a visual representation of the NFTs. Users log in via their wallet address and create listings that are eventually fulfilled on-chain.
2. **Premium membership** can be achieved by validating the user wallet address and providing exclusive content only to certain wallet addresses.
3. **Social media** can be anonymized by only using the users' wallet addresses as an identity while storing the data off-chain.

6.2. System Components

6.2.1. Ethereum-Based Wallet

Physical or software wallet used for signing the message received from the back-end server. Software wallets can be one of the following:

- MetaMask;
- CoinBase Wallet;
- TrustWallet;
- Exodus;
- Electrum.

Hardware wallets include:

- Ledger;
- Trezor;
- KeepKey.

6.2.2. Front-End

The graphical user interface is used to interact with the user wallet via browser-exposed methods. The technology stack is not relevant in our scenario, as we only use methods that software wallets expose to interact with the wallet [17]. However, robust frameworks should be preferred since the application can grow in complexity and size, and numerous libraries are available for such interactions.

6.2.3. Back-End

The back-end application should be secure and capable of recovering the message signed using the elliptic curve algorithm.

The elliptic curve digital signature algorithm (ECDSA) [18] is a variant of the digital signature algorithm (DSA) which uses elliptic curve cryptography.

Elliptic curve cryptography (ECC) [19] is a public key encryption algorithm based on elliptic curve mathematics. The main advantage of ECC is that it uses a smaller key length and provides a comparable level of security compared to the Rivest–Shamir–Adleman (RSA) encryption algorithm. ECDSA is a combination of ECC and DSA (digital signature algorithm).

Compared with RSA, the public key length of ECDSA is shorter, and the encrypted message will be smaller, so the computation and processing time will be shorter, and the memory and bandwidth requirements will be smaller.

Generation phase:
Select Ep(a,b), x, and $1 \leq x < n$.
Select G € Ep(a,b) with order n and compute Q = dG
Public key: (Ep(a,b), p,G,n,Q)
Private Key: x

Signature Algorithm
Select k, $1 \leq$ k < q.
kG = (x1,y1), r = x1 (mod n)
s = k $-$ 1 (H(m) + xr) (mod n)
(r, s) is the Signature of m.

Verification Algorithm
w = s^{-1} (mod n)
u1 = H(m)w (mod n)
u2 = rw (mod n)
u1G + u2Q = (x2,y2),
v = x_2 (mod n)
v = r $\rightarrow$ accept the Signature [20].

6.2.4. Database

This is used to store users' login attempts and nonces created for one-time usage.

7. Implementing Web3.0 Authentication

7.1. Proposed Login Flow

We will assume our application works both with the web3 signature provided by the users' signature and with the username and password. The traditional approach uses a JWT to store the users' session information and that would also be applicable for logging in with user wallets.

7.1.1. Wallet Connect

Connecting the wallet to the dApp requires confirmation from the wallet that the website will be able to interact with it.

As seen in Figure 4, as a result of the wallet interaction, a popup is displayed in the users' browser, announcing that he needs to consent to log in.

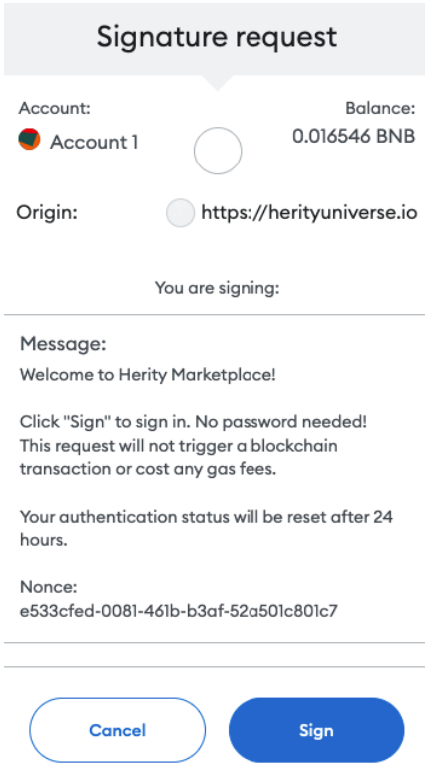


Figure 4. Wallet login using MetaMask.

7.1.2. Front-End Initiates Login Process

The first step of the login process is obtaining the nonce that has to subsequently be signed using the users' wallet. An HTTP request will be executed with the users's wallet address in order to fetch the nonce that has to be signed.

```
GET /auth/start
BODY:
{
address: '0xe3571d75004b8e5518..'
}
```

7.1.3. Back-End Generating the Nonce

Nonce (number used only once) will be generated from the back-end and stored in the database for future comparison.

```
begin
Read users' wallet address
Generate unique nonce
Store nonce/wallet address combination
Return nonce
end;
```

7.1.4. Front-End Signing the Nonce

The front-end layer retrieves the nonce which can either be numeric or in UUID (universal unique identifier) format and creates a signature message containing the nonce using the wallet address.

```
web3.eth.accounts.sign('c8b00baf−56be−450f−8a85−c05401f5a151', '0
    x4c0883a69102937d6231471b5dbb6204fe5129617082792ae468d01a3f362318');
> {
message: 'c8b00baf−56be−450f−8a85−c05401f5a151',
messageHash: '0x1da44b586eb0729ff70a73c326926f6ed5a25f
5b056e7f47fbc6e58d86871655',
v: '0x1c',
r: '0xb91467e570a6466aa9e9876cbcd013baba02900
b8979d43fe208a4a4f339f5fd',
s: '0x6007e74cd82e037b800186422fc2da167c747e
f045e5d18a5f5d4300f8e1a029',
signature: '0xb91467e570a6466aa9e9876cbcd 013
    baba02900b8979d43fe208a4a4f339f5fd6007
e74cd82e037b800186422fc2da167c747ef045e5d
18a5f5d4300f8e1a0291c'
}
```

The signed message is being sent to the back-end via a POST HTTP request, along with the initial nonce and the users' wallet address in order to be decrypted and validated.

```
POST /auth/complete

BODY:
{
"address":"0xe3571d75004b8e5518820958a57fccfc8a2d82ee",
"nonce":"c8b00baf−56be−450f−8a85−c05401f5a151",
"signature":"0xd8d90f7a1f36e5bc3a938091374a52
9d199498d8163127cecbd6eb3d4913dadd2d9e4d9373
01550e1128044d16fec092b7063d5b5ac5edfe5b2f7b
53b2ad99cf1b"
}
```

7.1.5. Back-End Decrypts the Signed Message

The signed message is then decrypted by the back-end in order to validate its authenticity and checks its correlation with the users' wallet address. If a user account exists for that wallet address, then a JWT is created based on that users' identity. If not, a new account is created, and a JWT for that account is created.

```
begin
Read nonce, wallet address, signed message
Retrieve the public key from the signed message using the nonce
Compare the public key with the wallet address
if(recoveredAddress = walletAddress)
Login successful
Mark login attempt
Return JWT
else
Login failed
Mark login attempt
Return error message
end;
```

7.1.6. Front-End Receives JWT

JWT is stored on the front end for future requests. As the user identity has been validated, all the subsequent requests made with that JWT will certify that the requester is the owner of the wallet address. This approach has potential setbacks as JWTs that do not have an expiry date are potentially vulnerable to a man-in-the-middle attack.

User privileges and identity can be reinforced by requiring the user to validate their identity using their wallet on specific, more-sensitive operations, such as changing private information or accessing sensitive content.

7.2. Authentication Flow Diagram

Authenticating using the users' wallet requires a multi-step process that should be completed in a matter of milliseconds. The need for knowing any information regarding the users' identity is no longer necessary, and the back-end heavily relies on signatures provided by third-party software that the user installs.

As opposed to traditional MFA mechanisms, authentication using software or hardware wallets can be done seamlessly from the convenience of a browser or a mobile phone.

Figure 5 outlines the end-to-end login flow using a third-party wallet software.

Communication between the front-end application and the backend is done via HTTPS, which is a secure communication protocol.

As seen in Figure 5, sending data between the front-end and the backend of our system in a secure and private way is critical. To ensure confidentiality and integrity, it is important to establish a secure connectivity. The use of HTTPS as a connection protocol is effective for achieving exactly this goal, as it is a widely used communication protocol that provides encryption for the data transmitted over the internet.

In the proposed implementation, all communication between the front-end and the back-end of the solution is done via HTTPS using built-in verbs and instructions. Encrypted communication channels are essential for protecting sensitive user data and preventing security breaches.

7.3. Security

Security in web3-related applications is achieved by making sure that the users are aware of the risks they are facing. Prevention helps in keeping the users' assets safe. It is also important to note that users have a part in safeguarding their assets since they must safeguard their private key or seed phrase and keep them in a secure location. If they are lost or stolen, they will be unable to access the assets.

To mitigate these threats, users must be cautious when disclosing personal information online and adopt safe mechanisms for storing and managing private keys and seed phrases, including hardware wallets. In addition, it is recommended to constantly double-check the URL of a website or sender email to ensure its legitimacy.

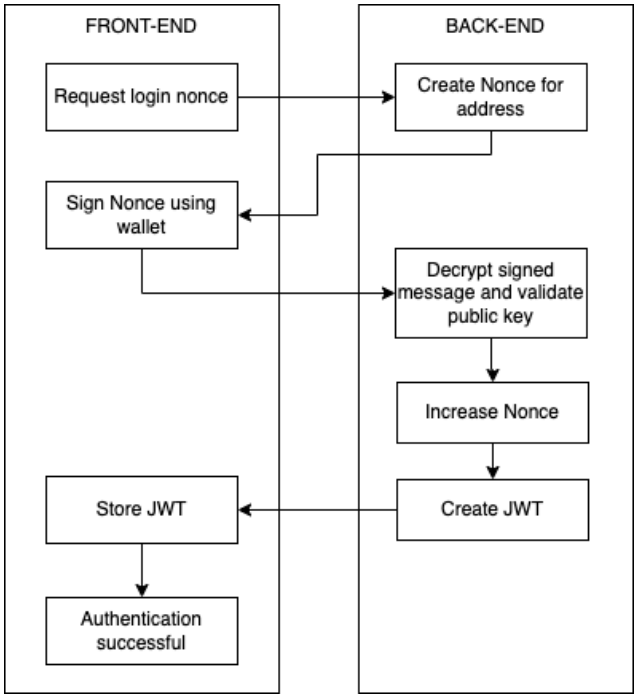


Figure 5. End to End wallet login flow.

Overall, Web3 authentication provides a safe technique for accessing decentralized apps and services, but users must secure their private keys and seed phrases to prevent any security breaches.

7.3.1. Security Attack Models

Phishing attacks, in which an attacker creates a fake website or email to trick a user into divulging their private key or seed phrase, are one of the primary threats to any blockchain-related service or authentication system.

Another risk is a "man-in-the-middle" (MITM) attack, where an attacker intercepts and alters the communication between a user and a legitimate website or service to steal private information or assets.

In a smart contract exploit, an attacker takes advantage of a flaw in the code of a smart contract to steal assets or alter the functionality of the contract.

Another example is a "51% attack" or "Sybil attack", where an attacker or a group of attackers control more than half of the mining power of a blockchain network, allowing them to reverse transactions, double-spend coins, and prevent other miners from adding new blocks to the chain. Most blockchain networks have built-in protection for Sybil attacks.

"Denial of service (DoS) attack" is another model where an attacker floods a network or website with a large number of requests to overload it and make it unavailable to legitimate users.

Though some of the models presented can be applied to traditional authentication mechanisms as well, the most popular attack model used in the decentralized application domain is phishing, where attackers trick users into revealing their private keys under the false pretense that their account has been locked or that the key is needed for extra security reasons.

7.3.2. Quantum Computing

Ethereum, like most blockchain technologies publicly available, is not intrinsically resistant to quantum computing. Quantum computers may have the ability to break several of the cryptographic techniques that are currently used to protect blockchain networks, including Ethereum's elliptic curve digital signature algorithm (ECDSA).

However, Ethereum's development community is working to address this issue by developing post-quantum cryptographic algorithms and implementing them in future versions of the Ethereum protocol. Additionally, Ethereum 2.0, which is currently being developed, is designed to be more resistant to quantum computing by using a different consensus mechanism called "proof-of-stake", which is believed to be more secure against quantum computers.

It is important to note that while the threat of a functional large-scale quantum computer is still uncertain, it is better to be proactive and take measures to protect the blockchain against potential quantum computing attacks.

8. Results

Implementing an authentication mechanism using blockchain wallets requires specialized knowledge of how blockchain works, web programming languages, cryptography, networking, and security. Technologies and frameworks used in the implementation are purely at the team's discretion. However, using the latest versions of packages and libraries is recommended to avoid any security breaches.

The proposed system was successfully implemented using Java and JavaScript, leveraging the power of Spring and Angular frameworks. The library used to interact with the wallet was Web3.js, as it benefits from the support of a large community and frequent security updates.

In terms of security, the proposed system has an increased vulnerability to the phishing attack model, which can only be prevented by exercising increased caution when logging in to unknown applications.

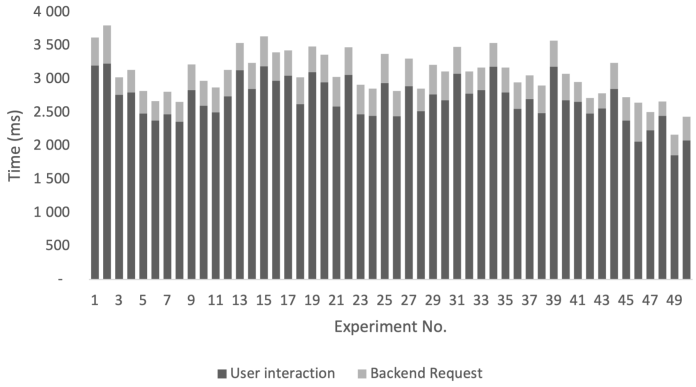
Having the proposed system implemented successfully, our study next aims to compare the times required for users to perform successful end-to-end login flows using software wallets under the form of a chrome extension and mobile application. Comparing these results with the results obtained from authentication with a similar one-factor authentication method using SMS codes can outline whether the proposed solution is faster (on top of other previously mentioned advantages).

To compare the proposed login mechanism with other relevant systems, a series of login operations was carried out (Figure 6). A relevant login model used as a comparison in our study is the SMS login. Rather than the conventional 2FA, SMS login only requires the user to confirm the login by entering the code received via SMS, without the need of entering any credentials. User reaction times and network request times have been measured in order to observe whether the proposed mechanism has a faster end-to-end login time compared to similar solutions.

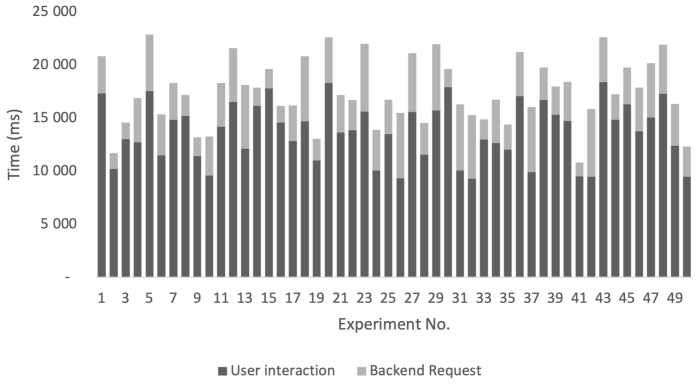
Running a number of 50 consecutive login operations using 3 distinct methods (wallet login, mobile wallet login, and SMS login), it has been observed that the average login time for a wallet login operation is 2.698 s (Figure 6a). The average network request is 375 ms, and it may vary based on network or server load. Similar to the wallet login method, the request times for a mobile wallet login operation have an average of 355 ms. Using a mobile wallet login, which depends on an external server for communication, has taken an average of 7294 ms (Figure 6b). This represents a 270% increase compared to the standard wallet login and the user reaction times have the most impact on the time (the user needs to scan a QR code and approve two distinct operations on the mobile phone).

Using a publicly available platform that has implemented a login mechanism using SMS codes, a similar number of tests have been run, and the average time spent in the operation is 13.799 s (Figure 6c). The network load for this experiment is significantly higher (3.782 s) due to the architecture choice of the observed platform. User interaction times are higher due to operations that need to be carried out (insert phone number, wait for SMS, insert SMS code, and wait for validation). An increase of 511%, respectively 189% compared to the standard wallet login time and mobile wallet login time was observed.

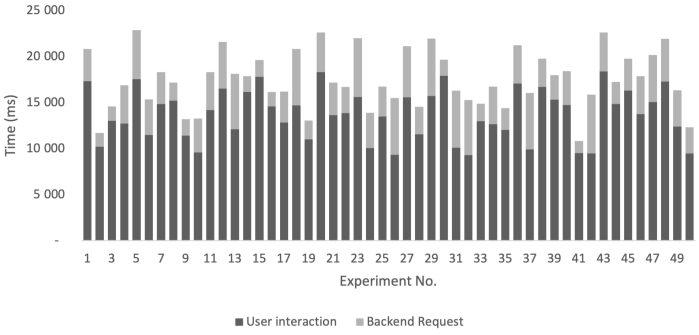
The expected time for performing an end-to-end login may vary depending on the network load, user reaction times, and implemented architecture. Nevertheless, the wallet login setup proposed appears to be faster compared to the other observed solutions.



(**a**)



(**b**)



(**c**)

Figure 6. Time spent on logging in. (**a**) Wallet login; (**b**) Mobile wallet login; (**c**) SMS login.

9. Discussion

The proposed solution was implemented successfully, and a growing number of decentralized applications that rely on the anonymity of the users' information have already implemented it (OpenSea, Rarible, Herity), which proves that the solution is mature enough to be used in other industries. The main disadvantage of the approach is that it requires specialized knowledge in areas such as blockchain and cryptography. The solution is vulnerable to phishing attacks, and since the identity is tightly linked to the public address, a user with a compromised private key might lose access to all his

assets. Support for implementing such solutions is still limited, as Web3 is not yet widely supported by mainstream websites and applications, which means that for the moment, use cases for the proposed mechanism are limited.

Nevertheless, there are multiple advantages that the proposed solution brings. One of them is decentralization. Since the authentication details are always in the users' control, there is no central entity to control and store credentials. Another advantage is that Web3 allows users to remain anonymous while still being able to prove their identity. This is particularly useful for people who want to protect their personal information from being misused or shared.

Results show that the solution has significantly lower end-to-end login times observed. Performing a log-in using a browser extension software wallet is five-fold when compared with the SMS login mechanism observed.

These challenges are an intriguing expansion for this study, and the authors want to broaden the outcomes of the current work by conducting a comprehensive investigation on blockchain data privacy and security.

10. Recommendations

When implementing a decentralized authentication mechanism based on Web 3.0, it is always important to evaluate whether the solution is suitable for the desired audience. There might be the case that the platform users have no blockchain knowledge or wallets and for this specific scenario, users might get confused and stop using the application (if already existent) or not start using it at all.

Web 3.0 authentication provides a synergy between existing on-chain resources and off-chain systems; thus, the implementation of such a system would most likely make sense in an application that either interacts with blockchain data or structures. Platforms implementing this authentication mechanism also need to consider whether they will rely on the complete anonymity of user information or will require an extra identity validation step.

Since the information is currently limited regarding such authentication mechanisms, their pitfalls and advantages, platforms implementing it must exercise caution and think of disaster recovery mechanisms for when a specific account is compromised.

11. Conclusions

As owning a digital wallet address gains in popularity, so does the need to accept user identities as anonymous without storing any user-related information. Authenticating using a wallet address opens the world for blockchain operations being performed both on-chain and off-chain.

Decentralized applications that rely on off-chain computation can only be accessed using an anonymous wallet address. However, the identities of the users holding some accounts must be validated using a KYC process for more sensitive financial operations.

Implementing a wallet authentication can be beneficial, as it can play the role of two-factor authentication for already existing systems that can be updated to support this mechanism. Moreover, systems that rely on the complete anonymity of users' data and information can rely on wallet authentication to provide exclusive content to specific users without ever interacting with the end user.

Therefore, this paper describes a straightforward implementation of user authentication using a novel browser interaction mechanism with users' software and hardware wallet. It is essential to mention that decentralized applications are gaining in popularity; sooner rather than later, authenticating users and providing them access to specific content will become mainstream.

Web3 on top of blockchain technology offers a balance of speed, security, and decentralization, allowing for secure and efficient transactions without the need for a centralized authority. Multiple types of decentralized applications heavily rely on providing specific content to anonymous users by combining information accessible on-chain and off-chain.

A study was carried out to compare the login times of the proposed mechanism with a similar solution based on SMS codes. The results show that implementing the proposed login mechanism can be achieved successfully, and the login times are significantly faster compared to similar solutions. EVM-based blockchains are the main platforms for creating decentralized applications.

The number of mobile-phone software application providers is currently limited, and communication is mostly done using WalletLink and WalletConnect providers. Until more systems are made available or a standard is implemented, mobile-phone software wallets are heavily dependent on the current solutions for any interaction with the decentralized applications.

Author Contributions: Conceptualization, A.P.; Methodology, D.A.S.; Software, A.P.; Validation, B.P., M.F. and D.A.S.; Investigation, A.P.; Writing—review & editing, B.P., M.F. and D.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

2FA	2-Factor Authentication
GUI	Graphical User Interface
MFA	Multi-Factor Authentication
Dapp	Decentralized Application
EVM	Ethereum Virtual Machine
NFT	Non-Fungible Token
KYC	Know Your Customer
DAO	Decentralized Autonomous Organization
DEX	Decentralized Exchange
P2P	Peer to peer
DeFi	Decentralized Finance
POW	Proof of Work
DoS	Denial of Service

References

1. Buldas, A.; Draheim, D.; Gault, M.; Laanoja, R.; Nagumo, T.; Saarepera, M.; Shah, S.A.; Simm, J.; Steiner, J.; Tammet, T.; et al. An Ultra-Scalable Blockchain Platform for Universal Asset Tokenization: Design and Implementation. *IEEE Access* **2022**, *10*, 77284–77322. [CrossRef]
2. Hendler, J. Web 3.0 Emerging. *Computer* **2009**, *42*, 111–113. [CrossRef]
3. Dhulavvagol, P.; Bhajantri, V.; Totad, S. Blockchain Ethereum Clients Performance Analysis Considering E-Voting Application. *Procedia Comput. Sci.* **2020**, *167*, 2506–2515. [CrossRef]
4. Canessane, R.A.; Srinivasan, N.; Beuria, A.; Singh, A.; Kumar, B.M. Decentralised Applications Using Ethereum Blockchain. In Proceedings of the 2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), Chennai, India, 14–15 March 2019; Volume 1, pp. 75–79. [CrossRef]
5. Tsepeleva, R.; Korkhov, V. Building DeFi Applications Using Cross-Blockchain Interaction on the Wish Swap Platform. *Computers* **2022**, *11*, 99. [CrossRef]
6. Jung, H.; Jeong, D. Blockchain Implementation Method for Interoperability between CBDCs. *Future Internet* **2021**, *13*, 133. [CrossRef]
7. Karapapas, C.; Syros, G.; Pittaras, I.; Polyzos, G.C. Decentralized NFT-based Evolvable Games. In Proceedings of the 2022 4th Conference on Blockchain Research and Applications for Innovative Networks and Services (BRAINS), Paris, France, 27–30 September 2022; pp. 67–74. [CrossRef]
8. Ding, W.; Hou, J.; Li, J.; Guo, C.; Qin, J.; Kozma, R.; Wang, F.Y. DeSci Based on Web3 and DAO: A Comprehensive Overview and Reference Model. *IEEE Trans. Comput. Soc. Syst.* **2022**, *9*, 1563–1573. [CrossRef]
9. Smith, J.; Nguyen, J. A Study of Ethereum Wallets and Their Security Measures. *Int. J. Blockchain Secur.* **2021**, *2*, 123–139.
10. Liao, C.H.; Guan, X.Q.; Cheng, J.H.; Yuan, S.M. Blockchain-based identity management and access control framework for open banking ecosystem. *Future Gener. Comput. Syst.* **2022**, *135*, 450–466. [CrossRef]

11. Ch, R.; Kumari D, J.; Gadekallu, T.R.; Iwendi, C. Distributed-Ledger-Based Blockchain Technology for Reliable Electronic Voting System with Statistical Analysis. *Electronics* **2022**, *11*, 3308. [CrossRef]
12. Imghoure, A.; El-Yahyaoui, A.; Omary, F. ECDSA-based certificateless conditional privacy-preserving authentication scheme in Vehicular Ad Hoc Network. *Veh. Commun.* **2022**, *37*, 100504. [CrossRef]
13. Ch, R.; Srivastava, G.; Reddy Gadekallu, T.; Maddikunta, P.K.R.; Bhattacharya, S. Security and privacy of UAV data using blockchain technology. *J. Inf. Secur. Appl.* **2020**, *55*, 102670. [CrossRef]
14. Song, J.G.; Kang, E.S.; Shin, H.W.; Jang, J.W. A smart contract-based p2p energy trading system with dynamic pricing on ethereum blockchain. *Sensors* **2021**, *21*, 1985. [CrossRef] [PubMed]
15. Liu, Z.; Xiang, Y.; Shi, J.; Gao, P.; Wang, H.; Xiao, X.; Wen, B.; Li, Q.; Hu, Y.C. Make Web3.0 Connected. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 2965–2981. [CrossRef]
16. Keizer, N.V.; Yang, F.; Psaras, I.; Pavlou, G. The Case for AI Based Web3 Reputation Systems. In Proceedings of the 2021 IFIP Networking Conference (IFIP Networking), Espoo and Helsinki, Finland, 21–24 June 2021; pp. 1–2. [CrossRef]
17. Gong, L.; Alghazzawi, D.M.; Cheng, L. BCoT sentry: A blockchain-based identity authentication framework for IoT devices. *Information* **2021**, *12*, 203. [CrossRef]
18. Nyame, G.; Qin, Z.; Obour Agyekum, K.O.B.; Sifah, E.B. An ECDSA approach to access control in knowledge management systems using blockchain. *Information* **2020**, *11*, 111. [CrossRef]
19. Chen, C.L.; Yang, J.; Tsaur, W.J.; Weng, W.; Wu, C.M.; Wei, X. Enterprise data sharing with privacy-preserved based on hyperledger fabric blockchain in IIOT's application. *Sensors* **2022**, *22*, 1146. [CrossRef] [PubMed]
20. Prabu, M.; Shanmugalakshmi, R. A Comparative Analysis of Signature Schemes in a New Approach of Variant on ECDSA. In Proceedings of the 2009 International Conference on Information and Multimedia Technology, Jeju, Republic of Korea, 16–18 December 2009; pp. 491–494. [CrossRef]