Towards a Single-Sign-On Authentication Architecture Based on OpenID Connect Protocol and Blockchain Technology

Assane Ilboudo(✉), Didier Bassole, Justin Pegdwindé Kouraogo, Gouayon Koala, and Oumarou Sie

Laboratoire de Mathématiques et d'Informatique, Université Joseph Ki-Zerbo, Ouagadougou, Burkina Faso
assaneilboudo1998@gmail.com
https://www.ujkz.bf

Abstract. OpenID Connect is a delegated authentication protocol used in web and mobile applications. It emphasizes the crucial role of third-party applications, also known as Relying Parties, which securely request information from an identity provider. In this article, we have proposed an approach based on the Ethereum blockchain to enhance the authentication process and user account privacy within the OpenID Connect protocol. This mechanism adds an additional layer of protection.

Keywords: Authentication · Security · Blockchain · OpenID Connect

1 Introduction

Nowadays, the number of internet users continues to grow globally. According to estimates from the second half of 2022, based on global Internet usage statistics, there are now over 4.95 billion active Internet users[1] among the eight (08) billion inhabitants of the planet[2]. This significant increase can be attributed to the growing proliferation of online services and remarkable advancements in network technologies [1]. However, this expansion comes with a proliferation of identification data for each user, exacerbating the inherent risk of losing or forgetting this information.

In this context, the adoption of Single Sign-On (SSO) systems emerges as a relevant solution to address various issues associated with user authentication on the web. The single sign-on system offers practical advantages by allowing users to access a multitude of applications with a single authentication [2]. Users only need to remember one set of credentials to access various services, thereby reducing cognitive load and mitigating the frustration associated with managing multiple passwords.

[1] https://fr.statista.com/statistiques/985232/nombre-utilisateurs-internet-monde/.
[2] https://www.un.org/fr/global-issues/population.

K. Cheikh M. F. Kebe et al. (Eds.): InterSol 2024, LNICST 610, pp. 87–99, 2025.
https://doi.org/10.1007/978-3-031-86493-3_8

In the single sign-on system, each user has an account managed by an Identity Provider (IdP) [3]. When a user logs into a service, they are redirected to the identity provider to authenticate. The IdP serves as a trusted third party for service providers and users. However, the identity provider may be vulnerable to certain attacks, such as sophisticated phishing attacks and identity theft, leading to data leaks and compromising users' privacy [2,4,5].

Furthermore, single sign-on authentication also poses potential security issues, including the risk of session hijacking and authentication token interception [3,6–8]. The robustness and confidentiality challenges in single sign-on authentication systems are currently a major concern for the computer security research community.

The main goal of this work is to enhance the robustness of the authentication process within the OpenID Connect protocol by proposing a secure architecture that integrates a decentralized multi-factor authentication mechanism based on the Ethereum blockchain.

The remainder of this article is organized as follows: in Sect. 2, we discuss the related works; in Sect. 3, we conduct a comparative study of different SSO protocols. Section 4 discusses our methodological approach and we conclude this work in Sect. 5.

2 Related Works

The widely used Single Sign-On protocols nowadays include OpenID Connect, supported by major digital companies such as Microsoft, Amazon, Google, Meta, PayPal, Verizon, Salesforce, Oracle, VMWare, IBM, WordPress, Yahoo, GitHub, and Twitter. OpenID Connect is also considered the new standard for Single Sign-On systems [9,10]. According to Zhang et al. [11], over a million websites support Single Sign-On (SSO) through OpenID Connect. Despite its widespread adoption on the web, mobile devices, cloud, Internet of Things (IoT), and SSO systems, OpenID Connect has a weakness. It inadequately verifies authenticity, merely validating the parameters of a request without confirming the identity of the request sender. Consequently, attackers can gain unauthorized access to restricted resources by exploiting theft, insufficient protection, and/or incomplete token validation.

To address these vulnerabilities, Yousra et al. [12] proposed a new model aimed at enhancing the security of OpenID Connect using blockchain technology and non-fungible tokens. Despite these improvements, it is important to note that vulnerabilities still exist. For example, if a legitimate user's credentials (username and password) fall into the hands of an attacker, they can use them to log in without the system effectively detecting the impersonation of the legitimate user's identity. This illustrated deficiency needs to be examined to strengthen the security of the conventional authentication process, which encompasses traditional methods such as the use of usernames and passwords, one-time codes, or security questions. These methods are employed to verify a user's identity before granting access to a system or information. Thus, how can the robustness of the

authentication process within the OpenID Connect protocol be strengthened? Yuki Ezawa et al. [13] presented an authentication system based on a blockchain-based PKI structure and smart contracts via a verification and authentication server, providing enhanced security during identity authentication. However, it is important to note that this approach requires the user to enter sensitive information such as the public key certificate, a random number, and a signature on the mobile login screen. In case of mobile loss, the user will no longer be able to authenticate, as the mobile is necessary to generate the required key certificate for authentication. In 2020, Mustafa [14] developed a Single Sign-On authentication system based on a private blockchain. In this system, each user has a pair of public and private addresses defined on the private blockchain network. This system relies on the OAuth2 protocol and uses a two-factor authentication method via a specifically developed mobile application, enhancing user information security. However, the creation of the public and private address pair via a mobile application before logging into a website poses a challenge. In case of mobile loss, the user will no longer be able to authenticate, as the mobile is necessary to generate the required key certificate for authentication. Also, in 2023, Shuhan et al. [15] introduced a federated identity system based on blockchain technology. Federated identity systems based on SAML face centralization issues by depending on a single identity provider within the federation, creating a single point of failure. To address this problem, the authors opted for a decentralized model based on the blockchain. In 2023, Yawalkar et al. [16] developed a federated identification and audit system based on blockchain technology. This system, relying on the SAML protocol, offers a decentralized approach to grant and control federated identities, allowing users to access online services provided by marketplace business partners. The framework is powered by smart contracts executed on the blockchain, ensuring secure and transparent management of federated identity creation and validation processes. Additionally, the system integrates an audit mechanism, enabling users and business partners to closely monitor all activities associated with the use of federated identities. The fundamental goal of this research is to provide a robust and effective solution for managing identities and transactions on a marketplace platform, leveraging the benefits of blockchain to ensure the integrity and efficiency of the process. In 2020, Nikos Fotiou et al. [17] proposed the creation of a new type of OAuth 2.0 token backed by distributed ledger technology, specifically blockchain. This proposal aims to enhance the security and management of OAuth 2.0 tokens used in online authorization protocols. This solution leverages Ethereum to store information for audits and verify token integrity. Ethereum smart contracts simplify revocation, separation of authorization servers and resource servers, enable token delegation, and offer opportunities for token exchange for fungible assets. In 2023, Fugkeaw [18] introduced a D2-IAM (Decentralized and Distributed Identity and Access Management) system based on blockchain supporting SSO authentication, authorization, and preventive access control with a supervisor in the cloud computing domain. All these features are orchestrated by a series of smart contracts, resulting in more autonomous and traceable access control procedures.

SSO authentication relies on token usage, reducing communication overhead between multiple systems. The D2-IAM system is based on OAuth2.

3 Comparative Study of Single Sign-On Protocols

3.1 Criteria and Comparison

In this section, we will conduct a comparative study of the protocols used in Single Sign-On systems, based on the following criteria (Table 1):

- Authorization and Authentication
- Token Format
- Data Integrity/Non-Repudiation
- Data Confidentiality/Privacy
- Support for Web Applications and Native Mobile Applications
- Lightweight Standard/Protocol

Table 1. Comparative analysis of Single Sign-On protocols

Criteria	SAML [19]	OAuth2 [19]	OIDC [19]
Authorization and Authentication	Mainly focused on authentication, with some authorization features	Emphasizes authorization, does not provide authentication	Combines authentication with the OAuth2 authorization framework
Token Format	XML	XML, JSON, JWT	JSON, JWT
Data Integrity/Non-Repudiation	Ensures data integrity with XML signatures	Security depends on the underlying transport (e.g., HTTPS)	Uses JSON Web Tokens (JWT) signatures to ensure data integrity
Data Confidentiality/Privacy	Assertions can be encrypted for confidentiality	Confidentiality depends on secure transport TLS	JWTs can be encrypted for confidentiality
Support for Web Applications and Native Mobile Applications	Primarily designed for web applications	Supports both web and native mobile applications	Supports both web and native mobile applications
Lightweight Standard/Protocol	Heavy due to the use of XML	Lighter	Lighter

In order to enhance the authentication process within the Single Sign-On system, some researchers, such as Mustafa [14] and Ezawa [13], have opted for a second-factor authentication (2FA) based on an application installed on a smartphone. However, in the event of a lost smartphone, the user would be unable to access their account. Similarly, in the case of a smartphone being stolen by a malicious individual, it could be used to impersonate the legitimate user and gain access to their account.

In contrast, author Yousra [12] did not integrate a second-factor authentication (2FA) into their Single Sign-On authentication system. Given that in Single Sign-On systems, the user relies on a single set of credentials, namely their username and password, Yousra's system presents an increased risk in the event of credential theft by a malicious actor, potentially compromising access to all accounts.

We have chosen the OpenID Connect protocol, in line with Yousra's prior use of this protocol, who also integrated blockchain technology into their Single Sign-On system. To enrich our contribution, we plan to build upon Yousra's advancements by incorporating a second-factor authentication (2FA) mechanism. This mechanism will be based on blockchain technology, and the user will be able to perform the second-factor authentication (2FA) using either their smartphone, their computer, or both, thus providing increased flexibility.

3.2 Description of the Operation of OpenID Connect

Operation of the OpenID Connect Protocol

OpenID Connect is a widely used authentication and authorization protocol that enables users to securely authenticate across various online applications and services. It is built on OAuth 2.0 [17], an authorization protocol, and adds an authentication layer, making it a popular choice for modern authentication systems. The OpenID Connect protocol offers three distinct flows to authenticate the end user with a Relying Party (RP): the authorization code flow, the implicit flow, and the hybrid flow. Among these three options, the authorization code flow is the most widely used for OpenID Connect protocol authentication due to its robustness and security.

To start using the services of the OpenID Connect protocol as an end user, the RP must first undergo a registration process with the Identity Provider (IdP). This registration involves configuring the necessary parameters and information to establish the connection between the RP, IdP, and the user. This registration process ensures a smooth and secure interaction among the parties involved in OpenID Connect authentication. It is designed to establish a standardized framework, allowing users to log in to multiple online services while preserving their security and privacy. The operation of the OpenID Connect protocol is illustrated in Fig. 1.

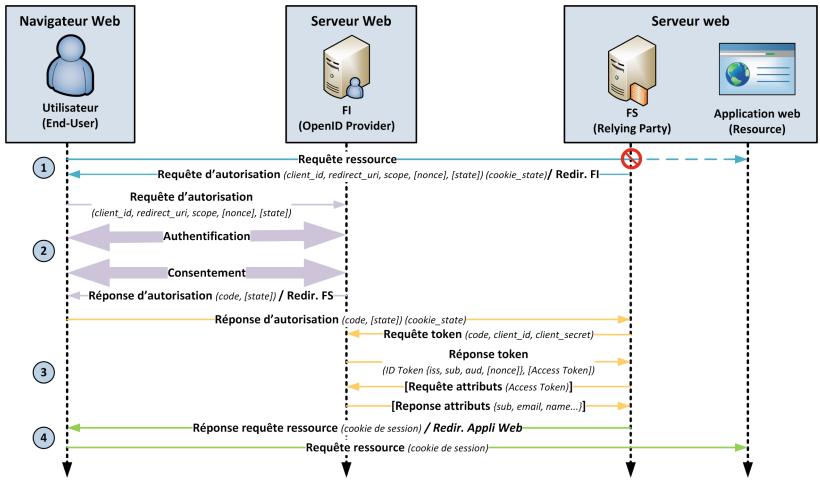

Fig. 1. Operation of the OpenID Connect protocol [20]

Some Parameters Used in the OpenID Connect Protocol Flow

- **client_id**: Unique identifier of the Service Provider with the Identity Provider. This is a public value, typically generated randomly by the Identity Provider,
- **client_secret**: This value is optional, but it is commonly used to authenticate the Service Provider when retrieving an authorization code from the Identity Provider. It must remain confidential and be known only to the two parties,
- **redirect_uri**: URL belonging to the domain of the Service Provider, used by the Identity Provider after user authentication to send the authorization response back to the Service Provider (via the user).
- **scope**: Lists the user's resources protected by the Identity Provider that the Service Provider wishes to access. This typically includes additional identity information such as email address.

Phase 1: An unauthenticated user seeks to access a protected resource. Their request is intercepted by the Service Provider, triggering an authorization request initiated by the Service Provider. This request is then sent back to the user, who presents it to their Identity Provider for validation.

Phase 2: The Identity Provider reviews the authorization request and prompts the user to log in to prove their identity. Once authenticated, the user is prompted to give consent to share the information required by the Service Provider (SP), defined in the "scope" parameter. After obtaining user consent, the Identity Provider sends an authorization response to the Service Provider (SP) via the user's browser. This response includes an essential authorization code.

Phase 3: The Service Provider (SP) verifies the authorization response submitted by the user and retrieves the authorization code. Subsequently, the service

system sends this code to the Identity Provider through a token request. This message is authenticated using the client credentials (client_ID and client_secret) and is transmitted via the token endpoint. The Identity Provider authenticates the Service Provider, verifies the authorization code, and in return sends the Service Provider the identity token (ID Token) corresponding to the authenticated user. At this stage, it is also possible for the Service Provider to obtain an access token to acquire additional identity information by sending an attribute request.

Phase 4: The Service Provider examines the ID Token, identifies the user, and performs a verification to control their access. If the user is authorized to use the requested service, the Service Provider finally grants them access to the initially requested resource. When a user wants to access secure data, a Relying Party (RP) requests approval from the Identity Provider (IdP) via the OpenID Connect protocol, but implementation errors can expose this request to attacks. Security must be strengthened by using TLS, following OpenID Connect, and blockchain technology can offer an additional layer of protection.

4 Methodological Approach

Our system is based on an existing reference model aimed at enhancing security and preserving privacy within OpenID Connect systems integrating the Ethereum blockchain and Non-Fungible Tokens (NFTs) [12]. This existing reference model takes a comprehensive approach, ensuring the integrity, availability, and confidentiality of OpenID Connect protocol security parameters, while limiting their access to authorized entities through a robust verification process.

By merging and expanding the definitions of Bal and Ner [21], Regner et al. [22], and Leech [23], we define a Non-Fungible Token (NFT) as "a cryptographically unique, indivisible, irreplaceable, and verifiable token representing a given asset, whether digital or physical, on a blockchain." Currently, the vast majority of NFTs are built on the Ethereum blockchain network and are therefore Ethereum tokens. For example, suppose we want to acquire gold without becoming its owner [23]. Someone could create a token whose value changes based on the price of gold. Instead of physically owning gold, we hold a representation in the form of tokens. This means that we do not own tangible, fungible gold, but rather a representation through tokens. These tokens are considered more secure because it is much more difficult to hack an Ethereum token than to break into someone's physical possession. The blockchain enables this by being immutable; once a transaction is validated, it cannot be modified [24]. We use these NFTs to enhance the security of access tokens to the account.

4.1 Strengthening Authentication Through 2F

Trust in technology has been seriously compromised by the alarming rise in online fraud cases, phishing attacks, and cybercrime [25]. The rapid growth and increasing sophistication of cybercrime have rendered traditional authentication

mechanisms (username and password) ineffective in safeguarding users' private data. To address this situation, OTPs (One-Time Passwords) and single sign-on systems have emerged as preferred solutions to counteract these fraudulent activities [26–28]. OTP represents a type of two-factor authentication mechanism. Simply put, an OTP is a time-based access token provided by the bank or application, containing numerical or alphanumeric values.

Various means are employed to deliver OTPs to users, such as software tokens through mobile applications, hardware tokens (like key fobs), and on-demand OTPs via SMS or emails [29]. Before completing the transaction, the user is prompted to enter the OTP received on their registered phone number, email address, or both. Once the correct OTP is entered by the user, the transaction is finalized. However, even with the OTP mechanism, the possibility of SIM cloning persists, allowing fraudulent access to the OTP and the execution of transactions on behalf of the user [30]. Another conceivable scenario is the loss of the user's phone, resulting in the inability to receive the OTP and consequently, the impossibility of completing the transaction [31].

To address this issue encountered by existing MFA (Multi-Factor Authentication) systems, we have decided to integrate a decentralized multi-factor authentication mechanism into our approach. Indeed, this multi-factor authentication mechanism will be based on smart contracts. The user will receive a transaction via their wallet, which they must confirm by signing it with their private key. Then, this transaction will be sent back to the smart contract for verification. If the transaction is valid, the user's identity is confirmed. The wallet can be installed either on the computer's browser, the smartphone, or both, i.e., the computer's browser and the smartphone.

4.2 Integration of a Decentralized Multi-factor Authentication Mechanism into the Reference Model

To address the Broken En-User Authentication attack issue faced by the reference model, we are incorporating a decentralized multi-factor authentication mechanism within it, as illustrated in Fig. 2. By integrating a multi-factor authentication mechanism, our model specifically tackles the challenge posed by the previously mentioned Broken En-User Authentication attacks. By requiring multiple factors for authentication, we significantly complicate the task for attackers attempting to compromise a user's identity. Moreover, the decentralization of this mechanism ensures that user identity validation is not dependent on a single central point. This approach significantly enhances the system's resilience by minimizing risks associated with isolated vulnerabilities or attacks directed at a centralized point.

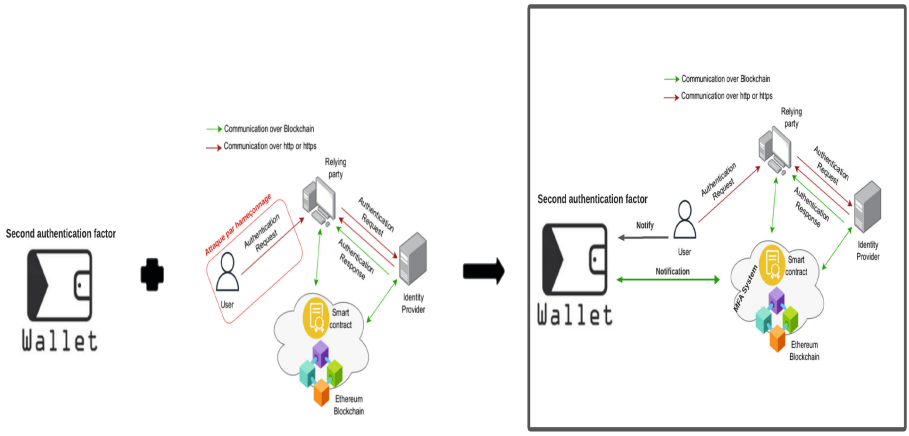

Fig. 2. Integration of a decentralized multi-factor authentication mechanism into the reference model

4.3 Proposed Model

In this subsection, we propose our model depicted in Fig. 3. Our approach involves three main actors:

- **the User**: the person who wishes to access an application or service and whose identity needs to be authenticated,
- **the Identity Provider (IdP)**: the entity that verifies the user's identity and issues authentication tokens and profile information to the Relying Party,
- **the Relying Party (RP)**: the application or service that requests and uses the user's identity information provided by the Identity Provider to grant access to its resources.

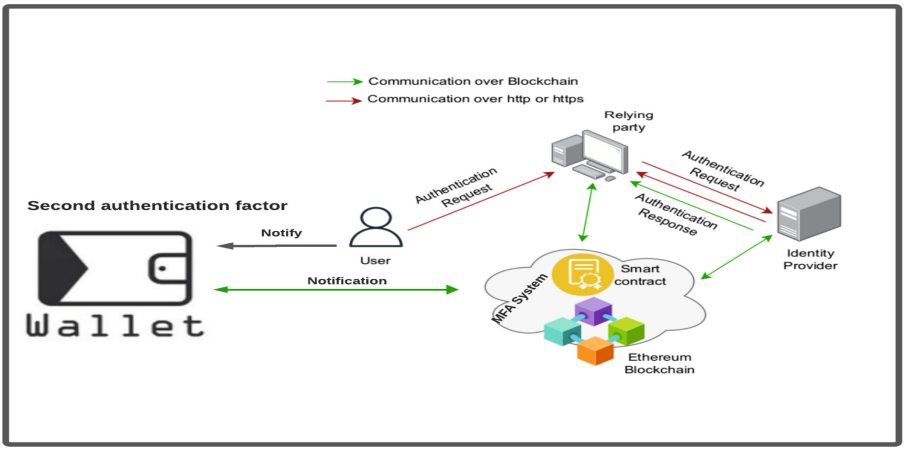

Fig. 3. Description of our methodological approach

Each of these actors assumes a dual role as an Ethereum node within the blockchain network. Specifically, the identity provider is responsible for managing the smart contract that generates non-fungible tokens (NFTs). When a user attempts to access a protected resource from the service provider requiring authentication, the process is triggered. The service provider (RP) initiates an authentication request with its identity provider (IdP), seeking the user's authorization to access their identity information stored with their IdP. This request is made by redirecting the user's browser to their identity provider, an entity that manages and provides credential information to authenticate users within a computer system. The identity provider reviews the authorization request and performs verification of the legitimacy of the service provider using the Ethereum blockchain-based smart contract technology.

If everything is verified, it prompts the user to log in to prove their identity using their credential data as the first factor of authentication. Next, the second factor of authentication comes into play, wherein the identity provider calls the smart contract on the Ethereum blockchain to verify the user's identity. The smart contract generates a specific transaction, which is then sent to the wallet for signing. At this point, a pop-up window appears, prompting the user to confirm the transaction by signing it using their private key. The smart contract then performs a verification using the public key associated with the user's Ethereum address to validate the signature provided by the wallet. If the signature is deemed valid, the contract is then authorized to perform the requested action. This process serves as the second factor of authentication, complementing the user's credential information, conclusively demonstrating that they are the authorized person to access the account.

This dual authentication enhances the security of the process because even if an attacker obtains the user's credential information, they cannot authenticate without confirming the wallet transaction. Once authenticated, the user is prompted to give consent to share the necessary information with the service provider. The identity provider creates tokens and calls on the smart contract to generate corresponding non-fungible tokens (NFTs) for each token. These NFTs are securely transmitted on the Ethereum blockchain to the recipient entity. An NFT is a form of unique digital token that represents ownership or proof of authenticity of a specific asset.

5 Conclusion

In this article, we have undertaken the persistent challenge of enhancing authentication within the OpenID Connect protocol by integrating Ethereum blockchain technology. By identifying existing vulnerabilities and exploring the limitations of current authentication systems, we have devised a robust multi-factor authentication mechanism using Ethereum blockchain technology. This mechanism adds an additional layer of protection within the OpenID Connect protocol, significantly reinforcing its security.

References

1. Ali, S.T.: MO-Auth: a novel approach for authentication in modern applications. Int. J. Adv. Res. Ideas Innov. Technol. **9**(2) (2023). www.IJARIIT.com
2. Mittal, N., Misbahuddin, M., Mustafa, A.S.: Enabling trust in single sign-on using DNS based authentication of named entities. Int. J. Microw. Wirel. Technol. **1**(1), 41–53 (2022). https://doi.org/10.5815/ijwmt.2022.01.05. http://www.mecs-press.org/ijwmt/ijwmt-v12-n1/v12n1-5.html
3. Ghasemisharif, M., Kanich, C., Polakis, J.: Towards automated auditing for account and session management flaws in single sign-on deployments. In: 2022 IEEE Symposium on Security and Privacy (SP), pp. 1774–1790. IEEE (2022). https://doi.org/10.1109/SP46214.2022.9833753. https://ieeexplore.ieee.org/abstract/document/9833753/
4. Sharif, M.H.U.: Web attacks analysis and mitigation techniques. Int. J. Eng. Res. Technol. 10–12 (2022). https://www.academia.edu/download/80622531/Web_Attacks_Analysis_and_Mitigation_Techniques.pdf
5. Xu, R., Yang, S., Zhang, F., Fang, Z.: MISO: legacy-compatible privacy-preserving single sign-on using trusted execution environments. In: 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P), pp. 352–372. IEEE (2023). https://ieeexplore.ieee.org/abstract/document/10190538/
6. Joseph, J., Bhadauria, S.: Cookie based protocol to defend malicious browser extensions. In: 2019 International Carnahan Conference on Security Technology (ICCST), pp. 1–6. IEEE (2019). https://doi.org/10.1109/CCST.2019.8888425. https://ieeexplore.ieee.org/document/8888425/
7. Zhong, L.: A Survey of Prevent and Detect Access Control Vulnerabilities. arXiv preprint arXiv:2304.10600 (2023). https://arxiv.org/abs/2304.10600
8. Ghasemisharif, M., Ramesh, A., Checkoway, S., Kanich, C., Polakis, J.: O single sign-off, where art thou? An empirical analysis of single sign-on account hijacking and session management on the web. In: 27th USENIX Security Symposium (USENIX Security 2018), pp. 1475–1492 (2018). https://www.usenix.org/conference/usenixsecurity18/presentation/ghasemisharif
9. Singh, T., Meenakshi: Prevention of session hijacking using token and session id reset approach. Int. J. Inf. Technol. **12**, 781–788 (2020). https://doi.org/10.1007/s41870-020-00486-w
10. Fett, D., Küsters, R., Schmitz, G.: A comprehensive formal security analysis of OAuth 2.0. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1204–1215 (2016). https://dl.acm.org/doi/abs/10.1145/2976749.2978385
11. Zhang, Z., Król, M., Sonnino, A., Zhang, L., Rivière, E.: El passo: privacy-preserving, asynchronous single sign-on. arXiv preprint arXiv:2002.10289 (2020). https://arxiv.org/abs/2002.10289
12. Yousra, B., Yassine, S., Yassine, M., Said, S., Lo'ai, T., Salah, K.: A novel secure and privacy-preserving model for OpenID connect based on blockchain. IEEE Access (2023). https://doi.org/10.1109/ACCESS.2023.3292143. https://ieeexplore.ieee.org/abstract/document/10172179/
13. Ezawa, Y., et al.: Designing authentication and authorization system with blockchain. In: 2019 14th Asia Joint Conference on Information Security (AsiaJCIS), pp. 111–118. IEEE (2019). https://doi.org/10.1109/AsiaJCIS.2019.00006. https://ieeexplore.ieee.org/abstract/document/8826910/

14. Tanrıverdi, M.: Design and implementation of blockchain based single sign-on authentication system for web applications. Sakarya Univ. J. Comput. Inf. Sci. **3**(3), 343–354 (2020). https://doi.org/10.35377/saucis.03.03.757459. https://dergipark.org.tr/en/download/article-file/1167051
15. Shuhan, M.K.B., Hasnayeen, S.M., Das, T.K., Sakib, M.N., Ferdous, M.S.: Decentralised Identity Federations using Blockchain. arXiv e-prints, arXiv-2305 (2023). https://ui.adsabs.harvard.edu/abs/2023arXiv230500315K/abstract
16. Yawalkar, P.M., Paithankar, D.N., Pabale, A.R., Kolhe, R.V., William, P.: Integrated identity and auditing management using blockchain mechanism. Measur. Sens. **27**, 100732 (2023). https://doi.org/10.1016/j.measen.2023.100732. https://www.sciencedirect.com/science/article/pii/S2665917423000685
17. Fotiou, N., Pittaras, I., Siris, V.A., Voulgaris, S., Polyzos, G.C.: OAuth 2.0 authorization using blockchain-based tokens. arXiv preprint arXiv:2001.10461 (2020). https://arxiv.org/abs/2001.10461
18. Fugkeaw, S.: Achieving decentralized and dynamic SSO-identity access management system for multi-application outsourced in cloud. IEEE Access **11**, 25480–25491 (2023). https://doi.org/10.1109/ACCESS.2023.3255885. https://ieeexplore.ieee.org/abstract/document/10066292/
19. Naik, N., Jenkins, P.: Securing digital identities in the cloud by selecting an apposite federated identity management from SAML, OAuth and OpenID connect. In: 2017 11th International Conference on Research Challenges in Information Science (RCIS), pp. 163–174. IEEE (2017). https://ieeexplore.ieee.org/abstract/document/7956534/
20. Rémi, C.C., Olivier, L.: OpenID Connect: présentation du protocole et étude de l'attaque Broken End-User Authentication. J. MISC (2018). http://paperstreet.picty.org/yeye/2018/magazine-misc-CassamChenaiL18/
21. Bal, M., Ner, C.: NFTracer: a Non-Fungible token tracking proof-of-concept using Hyperledger Fabric. arXiv preprint arXiv:1905.04795 (2019). https://arxiv.org/abs/1905.04795
22. Regner, F., Urbach, N., Schweizer, A.: NFTs in practice-non-fungible tokens as core component of a blockchain-based event ticketing application (2019). https://www.fim-rc.de/Paperbibliothek/Veroeffentlicht/1045/wi-1045.pdf
23. Leech, O.: What Are NFTs and How Do They Work? (2023). https://www.coindesk.com/what-are-nfts
24. Leloup, L.: Blockchain: La révolution de la confiance. Publisher Editions Eyrolles (2017). https://www.amazon.com/Blockchain-r%C3%A9volution-confiance-Laurent-Leloup/dp/2212566654
25. Huang, C.Y., Ma, S.P., Chen, K.T.: Using one-time passwords to prevent password phishing attacks. J. Netw. Comput. Appl. **34**(4), 1292–1301 (2011). https://www.sciencedirect.com/science/article/pii/S1084804511000427
26. Lamport, L.: Password authentication with insecure communication. Commun. ACM **24**(11), 770–772 (1981). https://doi.org/10.1145/358527.358533. https://dl.acm.org/doi/abs/10.1145/358790.358797
27. Lu, Z., Yu, H.: One Time Password Generating Method and Apparatus. U.S. Patent No. 8,184,872. U.S. Patent and Trademark Office, Washington, DC (2012). https://patents.google.com/patent/US8184872B2/en
28. Suriadi, S., Foo, E., Jøsang, A.: A user-centric federated single sign-on system. J. Netw. Comput. Appl. **32**(2), 388–401 (2009). https://www.sciencedirect.com/science/article/pii/S1084804508000519

29. Almuairfi, S., Veeraraghavan, P., Chilamkurti, N.: A novel image-based implicit password authentication system (IPAS) for mobile and non-mobile devices. Math. Comput. Model. **58**(1-2), 108–116 (2013). https://www.sciencedirect.com/science/article/pii/S0895717712001719
30. Wang, D., Wang, P.: Two birds with one stone: two-factor authentication with security beyond conventional bound. IEEE Trans. Dependable Secure Comput. **15**(4), 708–722 (2016). https://ieeexplore.ieee.org/abstract/document/7558124/
31. Huang, B., Khan, M.K., Wu, L., Muhaya, F.T.B., He, D.: An efficient remote user authentication with key agreement scheme using elliptic curve cryptography. J. Wirel. Pers. Commun. **85**, 225–240 (2015). https://doi.org/10.1007/s11277-015-2864-5