Modeling the Threats to Self-Sovereign Identities

Daniela Pöhn[1], Michael Grabatin[1], Wolfgang Hommel[1]

Abstract: Self-sovereign identity (SSI) is a relatively young identity management paradigm allowing digital identities to be managed in a user-centric, decentralized manner, often but not necessarily utilizing distributed ledger technologies. This emerging technology gets into the focus through the new electronic IDentification, Authentication and trust Services (eIDAS) regulation in Europe. As identity management involves the management and use of personally identifiable information, it is important to evaluate the threats to SSI. We apply the STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege) threat modeling approach to the core components of SSI architecture and the interactions between them. Based on the summarized results, we discuss relevant mitigation methods and future research areas.

Keywords: Self-sovereign identity; identity management; security analysis; threat modeling

1 Introduction

Online services ranging from news sites to shopping and banking are commonly used across the world in private and professional life. Originally, for each service, the user has to register and set up an account with a username and an authentication method, which is still most often just a password. Remembering these individual passwords is cumbersome for many users. Hence, either simple and, therefore, easy to crack passwords are chosen across several services by many users or password managers are applied, which also come in different qualities; for example, storing passwords in browsers often results in them being stored unencrypted on the user's device [OR20]. In order to reduce the burden on users and make the management of identities and credentials more efficient, cross-organizational identity management protocols, such as the Security Assertion Markup Language (SAML), Open Authentication (OAuth), and OpenID Connect (OIDC) were introduced already about 15 years ago. With these, users could set up one account at their home organization and use it across several internal and external services within defined trust boundaries. Currently, these home organizations are either universities for their members, companies for their employees, or major Internet companies, such as Google, Amazon, Meta, and Apple. Due to the architecture, these home organizations can collect detailed profile data about their users, e. g., when they log in to and, therefore, use which service. In order to give the user more or even full control over their data, self-sovereign identity (SSI) [Al17] has been proposed. SSI allows users (*holders*) to control the verifiable credentials (VCs) they hold typically

[1] Universität der Bundeswehr München, RI CODE, Werner-Heisenberg-Straße 39, 85577 Neubiberg, Germany {firstname.familyname}@unibw.de

doi:10.18420/OID2023_07

in a smartphone wallet and which they received from various organizations (*issuers*). The users then can present claims in form of verifiable presentations (VPs) to services (*verifiers*) depending on requested information and further decisions. The unique identifiers in this system are called decentralized identifiers (DIDs) and are typically anchored on a distributed ledger, although, e. g., public-key-infrastructure-based (PKI) SSI would be possible.

Even though the concept of SSI is comparably new and still merely applied in prototype or pilot projects, several organizations and larger projects work towards permanently establishing and integrating SSI into their environments. This includes the European Union, which explores SSI with their new version of the electronic IDentification, Authentication and trust Services (eIDAS) regulation [Eu22, Sh22]. As identity management involves personally identifiable information (PII), it is important to evaluate the security and data protection aspects of SSI. The lessons learned from projects focus more on organizational and technical aspects, see Mahula et al. [MTC21]. Few authors regard the threats to SSI and its components in detail. In consequence, we apply the common threat modeling technique STRIDE to evaluate the security of SSI and its elements. The contribution of this paper is threefold: 1) Evaluation of the threats to each architectural SSI element and its interactions, leading to an overall statement; 2) identification of relevant countermeasures; and 3) discussion of future work.

The remainder of this paper is structured as follows: In Section 2, we provide an overview of related work. Afterwards, in Section 3, we present our methodology. This is followed by the summarized results of threat modeling with STRIDE. In Section 4, we analyze the threats of each SSI component and interaction. We conclude the paper, discuss our approach, and provide an outlook to future work in Section 5.

2 Related Work

Only a few published approaches focus on the security respectively the threats of SSI. Naik et al. [NGJ21] propose an attack-tree-based risk analysis. Although the analysis is rather detailed, further elements such as trusted parties and threats, e. g., social engineering, are missing. Ahmed et al. [Ah22] provide several network-based attacks based on a literature review. This includes the 51% attack, Adversarial-in-the-Middle (AitM) attacks, impersonation attacks, replay attacks, Sybil attacks, forgery attacks, replacement attacks, collusion attacks, (Distributed) Denial-of-Service (DoS) attacks, and an outline of other attack types. Dingle et al. [Di19] focus on abusing verifiable credentials. In addition, uPort, Sovrin, and ShoCard are assessed [HECEK19, NJ20, NJ21]. Several authors analyze elements of SSI, such as wallets [Hu21, DS19], blockchain [Ch22, PP20, AK21, Sa20], and smart contracts [Hu19, Ku22]. This may be the case as Bitcoin and other decentralized financial systems such as decentralized finance (DeFi) are in use for recent years. Partly, countermeasures are included, which may be adapted for SSI use cases. One possible option within an SSI architecture is the usage of smart contracts. Ibba [Ib22] relates to the security of these among others. Last but not least, Wang et al. [WDZ18] analyze the security criteria of hash functions in blockchains. This shows that although several publications analyze elements used by SSI, almost no approaches analyze threats to SSI itself. In consequence, we analyze the threats by common threat modeling techniques.

3 Methodology

Threat modeling refers to various techniques to identify security and privacy-related design flaws and potential threats. Typical techniques include attack graphs and trees, STRIDE, and approaches by organizations, such as Open Web Application Security Project (OWASP) and MITRE ATTACK Cyber Kill Chain. For this paper, we choose STRIDE as it refers to the six main security threats of spoofing, tampering, repudiation, information disclosure, DoS, and elevation of privilege, which are timelessly important to applications and also directly related to identity management. STRIDE can be applied to each component and interaction to provide an overview of the threats to the infrastructure, as outlined by Shostack [Sh14]. In addition, we take aspects of UC-STRIDE [Da22], an approach for user-centric threat modeling into account. The SSI architecture is derived from research [GMM19, Li20, PH22] and Hyperledger [Hy17, Hy18] as the most prominent representative and consists of the layers, used by the roles issuer, holder, and verifier, with their agents and hubs. The evaluation, structured by component and interaction, leads to a discussion of the architecture.

Application Layer: For example, apps such as SSI wallets.

Credential Layer: Credential format, proof, revocation, exchange, and binding. Here, also smart contracts need to be considered.

Agent Layer: Communication, storage, and key management. For example, the nodes need to exchange messages in a shared ledger instance. Cryptographic functions, data repositories, application programming interfaces (APIs), and services are used here.

Public Trust Layer: This is the anchor of the trust, consisting of elements such as decentralized identifiers (DID) documents and resolution. These are, for the most common implementations today, anchored in a distributed ledger.

4 Threat Analysis of SSI Components and Interactions

Different threat actors could try to intentionally or could accidentally read, modify, or delete data or communication. A user might either directly try to access a service that they are not allowed to access or request resp. change claims they received in order to use these services. In addition, a user might make it impossible for other users to access a service, so they could alone profit from a service (e. g., a specific offer). Issuers might try to withhold data of users as they do not trust their ability to control it or verify more claims than possible for financial profit (e. g., a larger business share or paid service). A verifier could request more data than required for financial profit. Issuers and verifiers could try to act against competitors, for example, by making it impossible for them to communicate or verify claims. Outsider attackers could target all entities with different goals. The goals may include data, service disruption resp. usage, and other financial reasons. In consequence, the security of the SSI architecture with its components is important. We first analyze the the components of issuer and verifier, followed by the holder. Here, we take the architecture into account. Last but not least, distributed ledger usage as the prominent distributed storage is discussed.

4.1 Issuer and Verifier

The *issuer* generally consists of SSI-related and SSI-unrelated components, if it is not completely built on SSI. To access the DID network, the issuer applies software agents such as Hyperledger aries-cloudagent-python (ACA-py). The agent typically uses a web server, running a web application for the management, and a library to store and manage the keys. Depending on the type of ledger, either the DID is registered in the ledger or provides a service endpoint Uniform Resource Locator (URL). The latter is not typical for an issuer. Before issuing resp. verifying credentials, the issuer has to register the schema definition and the credential definition in the ledger. The schemas are searchable. For verifying credentials, two flows are possible: 1) either the holder requests them in a proposal or 2) the issuer sends an offer to the holder. Hyperledger Aries uses a Wallet Query Language (WQL) to fetch the credentials in the first place. The credentials might be stored in an LDAP server with interfaces to the SSI software. So far, the options for integrating LDAP and other traditional protocols are limited, as outlined by Kuperberg and Klemens [KK22]. To revoke credentials, Hyperledger applies a tails server with tails files. The software components of the *verifier* are similar. The web application of the verifier has a user (service) and admin (administration) login. The verifier specifies a schema, in which they want the claims.

Spoofing: The authenticity of the issuer resp. verifier and its users could be spoofed, either by insufficient authentication, insecure fallback authentication, no authentication at all, multiple accounts resp. nodes of the adversary, unencrypted communication, no integrity methods within the communication, or similar. If the access control resp. authentication is bypassed, an unauthorized person could modify data. Furthermore, holders need to be verified. In consequence, secure authentication methods, API security, and encryption mitigate most of these threats. In addition, the identities of computers, DID documents, data, and processes could be spoofed. This is possible by, for example, malware or stolen keys. Here, traditional security measures apply.

Tampering: The integrity of files/data (on a server or the used ledger), memory (server), network (ledger and off-chain), and sessions (either to other entities or administrators, to recalculate private keys for example) are targeted here. Especially private keys and claims are interesting, as they could be used to either generate financial benefits or receive access to other services. One option could be to misuse WQL or other querying languages. SSI software is built upon other components, such as web servers, organization-internal identity management systems, and databases. In addition, the claims could contain malicious data, for example, sent from the holder to the verifier, exploiting the database. In accordance with traditional network management, logging and monitoring are important measures to notice problems. This detection method though might need to be adapted for SSI. In addition, private keys and claims need to be secured, for example, by the principle of least privilege. Further common countermeasures apply here as well.

Repudiation: Non-repudiation is important for log files (actions performed by entities), credentials (not gained/requested), and blockchain-based ledgers (changes of blocks). Again, logging and monitoring are required. Due to the decentralized architecture of SSI, an exchange of threat information may help to identify problems at an early stage. Also, the registration of DIDs is important in this respect. Caused by the decentralized setting, outdated software becomes even more problematic. This shows that the organizational side of SSI needs further work.

Information disclosure: Confidentiality is required for the stored data (private keys, passwords, claims, PII), processes (by error messages, impersonation), data flows (blockchain, encryption, traffic flows), and other sources (social networks, etc.). This can be summarized as secure storage, cryptographic, secure design and programming including API security, and policies. Due to the decentralized architecture, the security of all participants is relevant as compromised entities could lead to information disclosure and modification. In consequence, some kind of trust level or flagging might be a way to signal trustworthiness.

Denial of Service: (D)DoS could target the availability of processes (memory, central processing unit (CPU), etc.), data stores (databases, identity management systems, blockchains), data flows (peer-to-peer or blockchain attacks), and interfaces (APIs). A simple way could be to request several verifiable credentials and not finish the workflow as expected by the protocol.

Elevation of privilege: Authorization relates to processes, missing/buggy user authorization checks, and data tampering (for example, by social engineering or malware). One way could be to target the API as it is the main way of communication. The web server with web application and database could be used to gain access if the configuration is not as intended or vulnerabilities exist. An indirect way of elevation is getting further credentials verified, which could have different reasons, such as financial gain, social engineering, compromise, incorrect verification, and spoofing/tampering.

4.2 Holder

The holder is the end-user of SSI with its wallet. In consequence, devices such as smartphones and computers and the SSI software running on them are important components. Many SSI implementations rely on users scanning quick response (QR) codes for interaction, which brings its own risks. Last but not least, the holders themselves have more control and, consequently, more responsibility. Hence, they must be regarded as well.

The *wallet* can be a smartphone app, a computer application, or even a cloud service. While cloud services provide a backup in the cloud and solve the problem of lost devices or credentials, they can pose an interesting target as seen with the LastPass incidents in 2022 [To22]. In theory, hardware wallets similar to cryptocurrencies might be used. Insider and included functionality (such as backdoors) are possible. Brute-force and social engineering attacks could target wallets. Therefore, the user interface should provide information regarding the trustworthiness and status of entities and SSI (e. g., showing that an entity is untrustworthy or that the entity and the name of the intended entity differ). Communication, stored data, and usage among others could lead to de-anonymization and information leakage. The *smartphone* is the typical device for SSI holders. These are prone to theft, malware, and remote access to manipulate or steal claims. In order to reduce the risk of the latter, a trusted execution environment (TEE) is often discussed. Although they tend to be tamper-resistant, they add an additional layer with possible vulnerabilities. In addition, their existence depends on the security model of the smartphone. As determined by the security model, different levels of trust and functionality could be enabled. Either way, the data stored on the smartphone needs encryption and proper authentication. If smartphones (and other end-user devices with a wallet) are shared with family members or friends, these guests should not have access to the wallet. This is typically enforced by additional accounts on the smartphone and strict separation of contexts. In consequence, public computers in internet cafes or hotels should not be used for SSI. Applying STRIDE, we see the following.

Spoofing: The authenticity relates to the authentication of the users directly or indirectly via its data or an API. When utilizing cloud services, other types of spoofing, such as link, frame, or URL spoofing could be used as well. In consequence, it is important to apply secure identification and authentication methods in addition to secure design and coding. If the wallet app is linked to a subscriber identity module (SIM) card, SIM card hijacking/swapping attacks and hence identity theft are further threats. In consequence, links to SIM cards might not be the best option. On the other hand, also email accounts can be taken over.

Tampering: The integrity of files/data, memory, and sessions are targeted here. As a result, the data has to be encrypted and use secure identification and authentication methods. In smartphones, TEE could be used. Hardware wallets could be stolen. Here, encryption and authentication are relevant. Either way, it is important not to let guest users access or interfere with the wallets. In consequence, also malware is a threat in this context (clipboard with information, malicious browser extensions, phishing, etc.). Similar to web applications, XSS and SQL injection could be used to modify data. Typical countermeasures should be applied also to smartphones.

Repudiation: The non-reputability is important for log files, credentials (not gained/requested), and blockchain (changes of blocks). As users are not used to seeing and working with log files, further work is required. If credentials are not gained or requested, a notification may provide a hint of malicious actions. In addition, the last log-in, actions, and changes of blocks could be noted.

Information disclosure: Confidentiality is required for the stored data (private keys, passwords, claims, PII), processes (by error messages, impersonation), data flows (blockchain, encryption, traffic flows), and other sources (social networks, etc.). The data could be stored in a TEE if the user utilizes a smartphone app. Furthermore, the application should come from a trustworthy source. As a negative example, in the Ethereum network, malicious clones of the popular MyEtherWallet wallet appeared. Updates should be provided as long as possible, as wallets are often open-source and vulnerabilities can be exploited in large-scale attacks. Data storage is important to protect claims and further information from unauthorized access. In consequence, authentication and authorization to access the app need to be enabled as well. The error messages while authenticating should not provide any information about the actual users or passwords. The messages have to be encrypted. Traffic flows could theoretically indicate the usage of certain SSI environments and services. Theoretically, dummy traffic could be added. Last but not least, the user should get educated to be aware of operations security and social engineering.

Denial of Service: (D)DoS could target the availability of processes, data stores, data flows, and interfaces. Offline variants should be explored to reduce the problems.

Elevation of privilege: Authorization relates to the permissions of users and processes. Wallets should apply the principle of least privilege and not permit resp. be permitted with root access. With cloud wallets, an escalation of privilege could lead to access to all wallets at the service. Hence, proper authentication and authorization in addition to monitoring and other further security measures are required on the server side.

Another issue with the holder is the *human* person, which could be tricked into different actions by the means of social engineering. Besides the before mentioned aspects, the holder could send claims to an untrusted verifier (e. g., trick-bot trojan, phishing mail, or other means) accept and send claims after several requests were made (similar to bypassing multi-factor authentication (MFA) by a request flood), or fail to notice address manipulation. Depending on the fallback methods, these could be misused either by external adversaries or people, the holder knows (in case of social recovery). *QR codes* are an easy way to redirect users to a specific website. Malicious QR codes could, similar to Averin and Zyulyarkina [AZ20], use structured query language (SQL) injection, cross-site scripting

(XSS), command injection, or social engineering to exploit the wallet or holder. In addition, maliciously crafted QR codes resp. websites could also target other verifiers. Further variations include fake QR generators and social engineering. Indirectly, the fingerprint of the wallet and smartphone could be generated and used for other purposes. Nevertheless, QR codes are an easy way to send users an URL and the problems are inherent. As long as no better solution is found, security measures should be applied before opening the URL. Here, further work is required. In consequence, awareness has to be raised. This could be accompanied by awareness methods, which are also future work.

4.3 Decentralized Ledger Technology

Although other technology, such as known public key infrastructure (PKI), could be utilized for SSI, most current approaches are built on distributed ledgers. Since distributed ledgers are also comparably new, leveraging the blockchain hype, we focus on this architecture and the underlying infrastructure. Important elements are the consensus mechanism and ledger, peer-to-peer network, network, and storage, and might include smart contracts.

Spoofing: Consensus mechanisms and ledgers generally have the threat of double spending, as also described for peer-to-peer networks. Saad et al. [Sa20] additionally describe consensus delay, forks, orphaned and stale blocks as well as block ingestion. Similar, selfish mining, precomputing, bribery attacks or feather forking, and collusion and time spoofing attacks could be possible, depending on the type of consensus mechanism. Thereby, blocks and forks of an adversarial could be favored. Blockchain, for example, has dependencies on databases. Micro-services are utilized, which have APIs. Further storage might be used as well, see smartphone wallets. In consequence, all these threats apply to SSI as well. This includes insufficient authentication at APIs, SQL injections, and local/remote file inclusion.

Tampering: According to Iqbal et al. [IM21], the Sybil attack enables breaking the consensus protocol, generating fake transactions, tampering with nodes' reputation, isolating nodes, modifying routing tables, and DDoS attacks. An adversary could include malicious data in the blocks (poisoning), targeting different entities or the SSI system. Last but not least, the cryptographic methods (hash algorithms, PKI, and key management) could be cracked or otherwise attacked [Ch22, WDZ18, Ib22, An22].

Repudiation: If all elements log and these log files cannot be changed by an adversarial, then non-reputability exist. As blockchain is not the only component, traditional methods need to be employed.

Information disclosure: Due to the different attacks, information could be disclosed.

Denial of Service: Peer-to-peer networks in general face several typical threats, summarized by Aggarwal et al. and Ahmed et al. [AK21, Ah22] as Finney attack, race attack, 51% attack, Sybil attack, eclipse attack, and additionally routing attack. The first attacks are possible if the adversary deploys or gains enough nodes to for example exclude others. Last but not least, (D)DoS attacks are possible within all elements of DLTs.

Elevation of privilege: Although DLTs and blockchain in general only store DIDs and other meta information, the implementation might be vulnerable.

Smart contracts can be used in SSI systems but do not have to. Known attacks on smart contracts include the decentralized autonomous organization (DAO) and parity attack in the Ethereum network. Also malicious contracts and unintentional vulnerabilities such as King of the Ether Throne, GovernMental, and Rock-Paper-Scissors happened. Putz and Pernul [PP20] describe re-entrancy attacks, delegate-call injection, injection of malicious code, and JavaScript Object Notation (JSON) injection vulnerabilities, among others. Tang et al. [Ta21] add integer over-/underflows, greedy contracts, DDoS, gas overspent if gas is the motivation, transaction ordering, prodigal contracts, vulnerable exception handling, destroyable contracts, authentication via external services, block timestamp dependencies, and unchecked and failed sending. In consequence, Huang et al. [Hu19] suggest secure design, implementation, testing, monitoring, and analysis.

Organizational aspects such as processes, trust, and governance need to be regarded. These are currently not established for SSI, although some approaches were proposed. Similar to identity federations, consortia could be established to form trust boundaries, or the Domain Name System (DNS) could be utilized [JJCK22]. In addition, the consensus mechanism needs to be evaluated for its threats and possible countermeasures such as bonus points for correct behavior or exclusion of members. The application process for future members could be circumvented by false input or social engineering. The exclusion or deactivation of untrustworthy members needs to follow certain rules. The update of the underlying infrastructure and software may require orchestration to reduce the attack surface. Last but not least, multiple adversaries could form groups to overrule the other members.

5 Conclusion and Outlook

With the recent advances of SSI and their uptake with eIDAS, we analyzed the threats to SSI based on their elements applying STRIDE. First, we described our methodology before applying it to the roles of the issuer, holder, verifier, and the element blockchain. In addition, we outline relevant mitigation methods. As a result, several threats exist. While traditional technology has several countermeasures to protect the infrastructure, some threats within SSI still need to be regarded in detail in future work. In this paper, we discussed threats based on STRIDE and summarized them based on the findings of our literature review. Although we identified several threats and mitigation, with more practical experience further threats might be added. To combat the threats, we gave a brief overview of mitigation methods. These need to be enhanced in future work. In contrast to traditional technologies such as PKI, where mitigation is common ground, blockchain and smart contracts are still comparably new and hence their security is still not fully explored. To the best of our knowledge, we found no publications focusing on the organizational aspects of SSI. Projects included the formulation of consortia similar to identity federations. This though does not include security management. Last but not least, the user gains more control of their data by applying the wallet. Hence, they need to take care of their data. Further research is required if users utilize their wallets in a privacy-preserving way, which awareness methods might help, and if there is any significant difference in design and implementation decisions, such as smartphone- or cloud-based wallets. In future work, we plan to explore organizational aspects related to the security of the decentralized infrastructure. In addition, we want to evaluate the threats to the existing implementations and different architecture variants. The security analysis will regularly be reevaluated and updated.

Bibliography

[Ah22] Ahmed, Md. Rayhan; Islam, A. K. M. Muzahidul; Shatabda, Swakkhar; Islam, Salekul: Blockchain-Based Identity Management System and Self-Sovereign Identity Ecosystem: A Comprehensive Survey. IEEE Access, 10:113436–113481, 2022.

[AK21] Aggarwal, Shubhani; Kumar, Neeraj: Chapter Twenty – Attacks on blockchain – Working model. In (Aggarwal, Shubhani; Kumar, Neeraj; Raj, Pethuru, eds): The Blockchain Technology for Secure and Smart Applications across Industry Verticals, volume 121 of Advances in Computers, pp. 399–410. Elsevier, Amsterdam, Niederlande, 2021.

[Al17] Allen, Christopher: , The Path to Self-Sovereign Identity. https://github.com/ChristopherA/self-sovereign-identity/blob/master/ThePathToSelf-SovereignIdentity.md, 2017. accessed May 10, 2023.

[An22] Ansar, Syed Anas; Arya, Swati; Aggrawal, Shruti; Yadav, Jaya; Pathak, Prabhash Chandra: Bitcoin-Blockchain Technology: Security Perspective. In: 3rd International Conference on Intelligent Engineering and Management (ICIEM). IEEE, pp. 291–296, 2022.

[AZ20] Averin, Andrey; Zyulyarkina, Natalya: Malicious Qr-Code Threats and Vulnerability of Blockchain. In: Global Smart Industry Conference (GloSIC). IEEE, pp. 82–86, 2020.

[Ch22] Chen, Xiaofeng; Wei, Zunbo; Jia, Xiangjuan; Zheng, Peiyu; Han, Mengwei; Yang, Xiaohu: Current Status and Prospects of Blockchain Security Standardization. In: 9th International Conference on Cyber Security and Cloud Computing (CSCloud)/8th International Conference on Edge Computing and Scalable Cloud (EdgeCom). IEEE, pp. 24–29, 2022.

[Da22] Datta, Prerit; Sartoli, Sara; Gutierrez, Luis Felipe; Abri, Faranak; Namin, Akbar Siami; Jones, Keith S.: A User-Centric Threat Model and Repository for Cyber Attacks. In: 37th ACM/SIGAPP Symposium on Applied Computing. SAC '22, p. 1341–1346, 2022.

[Di19] Dingle, P.; Hammann, S.; Hardman, D.; Winczewski, C.; Smith, S.: , Alice Attempts to Abuse a Verifiable Credential. https://github.com/WebOfTrustInfo/rwot9-prague/

`blob/master/final-documents/alice-attempts-abuse-verifiable-credential.pdf`, 2019. accessed May 10, 2023.

[DS19] Davenport, Amanda; Shetty, Sachin: Modeling Threat of Leaking Private Keys from Air-Gapped Blockchain Wallets. In: International Smart Cities Conference (ISC2). IEEE, pp. 9–13, 2019.

[Eu22] European Digital Identity – Digital Identity for all Europeans, accessed May 10, 2023.

[GMM19] Grüner, Andreas; Mühle, Alexander; Meinel, Christoph: An Integration Architecture to Enable Service Providers for Self-sovereign Identity. In: 18th International Symposium on Network Computing and Applications (NCA). pp. 1–5, 2019.

[HECEK19] Haddouti, Samia El; Ech-Cherif El Kettani, M. Dafir: Analysis of Identity Management Systems Using Blockchain Technology. In: International Conference on Advanced Communication Technologies and Networking (CommNet). IEEE, pp. 1–7, 2019.

[Hu19] Huang, Yongfeng; Bian, Yiyang; Li, Renpu; Zhao, J. Leon; Shi, Peizhong: Smart Contract Security: A Software Lifecycle Perspective. IEEE Access, 7:150184–150202, 2019.

[Hu21] Hu, Yiwen; Wang, Sihan; Tu, Guan-Hua; Xiao, Li; Xie, Tian; Lei, Xinyu; Li, Chi-Yu: Security Threats from Bitcoin Wallet Smartphone Applications: Vulnerabilities, Attacks, and Countermeasures. In: 11th ACM Conference on Data and Application Security and Privacy (CODASPY). p. 89–100, 2021.

[Hy17] Hyperledger Foundation: Hyperledger Architecture, Volume 1 – Introduction to Hyperledger Business Blockchain Design Philosophy and Consesus. Architecture paper, 2017.

[Hy18] Hyperledger Foundation: Hyperledger Architecture, Volume II – Smart Contracts. Architecture paper, 2018.

[Ib22] Ibba, Giacomo: A Smart Contracts Repository For Top Trending Contracts. In: 5th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB). IEEE/ACM, pp. 17–20, 2022.

[IM21] Iqbal, Mubashar; Matulevičius, Raimundas: Exploring Sybil and Double-Spending Risks in Blockchain Systems. IEEE Access, 9:76153–76177, 2021.

[JJCK22] Johnson Jeyakumar, Isaac H.; Chadwick, David W.; Kubach, Michael: A novel approach to establish trust in verifiable credential issuers in Self-sovereign identity ecosystems using TRAIN. In (Roßnagel, Heiko; Schunck, Christian H.; Mödersheim, Sebastian, eds): Open Identity Summit 2022. Gesellschaft für Informatik e.V., Bonn, pp. 27–38, 2022.

[KK22] Kuperberg, Michael; Klemens, Robin: Integration of Self-Sovereign Identity into Conventional Software using Established IAM Protocols: A Survey. In (Roßnagel, Heiko; Schunck, Christian H.; Mödersheim, Sebastian, eds): Open Identity Summit 2022. Gesellschaft für Informatik e.V., pp. 51–62, 2022.

[Ku22] Kushwaha, Satpal Singh; Joshi, Sandeep; Singh, Dilbag; Kaur, Manjit; Lee, Heung-No: Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract. IEEE Access, 10:6605–6621, 2022.

[Li20] Liu, Yue; Lu, Qinghua; Paik, Hye-Young; Xu, Xiwei; Chen, Shiping; Zhu, Liming: Design Pattern as a Service for Blockchain-Based Self-Sovereign Identity. IEEE Software, 37(5):30–36, 2020.

[MTC21] Mahula, Stanislav; Tan, Evrim; Crompvoets, Joep: With Blockchain or Not? Opportunities and Challenges of Self-Sovereign Identity Implementation in Public Administration: Lessons from the Belgian Case. In: DG.O2021: The 22nd Annual International Conference on Digital Government Research. ACM, p. 495–504, 2021.

[NGJ21] Naik, Nitin; Grace, Paul; Jenkins, Paul: An Attack Tree Based Risk Analysis Method for Investigating Attacks and Facilitating Their Mitigations in Self-Sovereign Identity. In: Symposium Series on Computational Intelligence (SSCI). IEEE, pp. 1–8, 2021.

[NJ20] Naik, Nitin; Jenkins, Paul: uPort Open-Source Identity Management System: An Assessment of Self-Sovereign Identity and User-Centric Data Platform Built on Blockchain. In: International Symposium on Systems Engineering (ISSE). IEEE, pp. 1–7, 2020.

[NJ21] Naik, Nitin; Jenkins, Paul: Sovrin Network for Decentralized Digital Identity: Analysing a Self-Sovereign Identity System Based on Distributed Ledger Technology. In: International Symposium on Systems Engineering (ISSE). IEEE, pp. 1–7, 2021.

[OR20] Oesch, Sean; Ruoti, Scott: That Was Then, This Is Now: A Security Evaluation of Password Generation, Storage, and Autofill in Browser-Based Password Managers. In: 29th USENIX Security Symposium. pp. 2165–2182, 2020.

[PH22] Pöhn, Daniela; Hommel, Wolfgang: Reference Service Model Framework for Identity Management. IEEE Access, 10:120984–121009, 2022.

[PP20] Putz, Benedikt; Pernul, Günther: Detecting Blockchain Security Threats. In: International Conference on Blockchain (Blockchain). IEEE, pp. 313–320, 2020.

[Sa20] Saad, Muhammad; Spaulding, Jeffrey; Njilla, Laurent; Kamhoua, Charles; Shetty, Sachin; Nyang, DaeHun; Mohaisen, David: Exploring the Attack Surface of Blockchain: A Comprehensive Survey. IEEE Communications Surveys & Tutorials, 22(3):1977–2008, 2020.

[Sh14] Shostack, Adam: STRIDE. In: Threat Modeling: Designing for Security. pp. 61–86, 2014.

[Sh22] Sharif, Amir; Ranzi, Matteo; Carbone, Roberto; Sciarretta, Giada; Ranise, Silvio: SoK: A Survey on Technological Trends for (Pre)Notified EIDAS Electronic Identity Schemes. In: 17th International Conference on Availability, Reliability and Security (ARES). ACM, 2022.

[Ta21] Tang, Xiangyan; Zhou, Ke; Cheng, Jieren; Li, Hui; Yuan, Yuming: The Vulnerabilities in Smart Contracts: A Survey. In (Sun, Xingming; Zhang, Xiaorui; Xia, Zhihua; Bertino, Elisa, eds): Advances in Artificial Intelligence and Security. Springer International Publishing, pp. 177–190, 2021.

[To22] Toubba, Karim: , Notice of Recent Security Incident. https://blog.lastpass.com/2022/12/notice-of-recent-security-incident/, 2022. accessed May 10, 2023.

[WDZ18] Wang, Maoning; Duan, Meijiao; Zhu, Jianming: Research on the Security Criteria of Hash Functions in the Blockchain. In: 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts (BCC). pp. 47–55, 2018.