applied sciences

Article

Analyzing the Threats to Blockchain-Based Self-Sovereign Identities by Conducting a Literature Survey

Daniela Pöhn *, Michael Grabatin and Wolfgang Hommel

Research Institute CODE, University of the Bundeswehr Munich, 85579 Neubiberg, Germany; michael.grabatin@unibw.de (M.G.); wolfgang.hommel@unibw.de (W.H.)
* Correspondence: daniela.poehn@unibw.de

Abstract: Self-sovereign identity (SSI) is a digital identity management model managed in a decentralized manner. It allows identity owners to manage and store their digital identities in a software wallet, for example, on a smartphone, without relying on centralized providers. This approach tries to enhance the security and privacy of digital identities and, thereby, their owners. With the new eIDAS regulation, elements of SSI, such as the wallet, are being pushed onto the market. However, since the model is relatively new, the security threats are still not fully known. This is shown by a brief security analysis of selected existing SSI wallets. In order to get a picture of the known threats, we systematically analyze and categorize related work in the field of SSI and elements applied by SSI. We then evaluate their application to current SSI systems and identify future work.

Keywords: self-sovereign identities; SSI; identity management; threat analysis; security; survey

check for updates

Citation: Pöhn, D.; Grabatin, M.; Hommel, W. Analyzing the Threats to Blockchain-Based Self-Sovereign Identities by Conducting a Literature Survey. *Appl. Sci.* **2024**, *14*, 139. https://doi.org/10.3390/app14010139

Academic Editor: Gianluca Lax

Received: 7 November 2023
Revised: 18 December 2023
Accepted: 20 December 2023
Published: 22 December 2023

1. Introduction

With the upcoming new electronic Identification, Authentication and Trust Services (eIDAS) regulation, changes to the current eID ecosystem in Europe will be made. eIDAS as of 2014 [1] focuses on the interoperability of identities within the member states of the European Union (EU) for the use case of eGovernment. However, with the new regulation [2], further use cases should be made possible, such as universities, health, finance, online services, and mobility. One major change is the introduction of a wallet [3], described in detail in the architecture and reference framework [4]. These wallets are typically utilized within self-sovereign identities (SSI) as a place to store user credentials and further data. With the use of wallets on a smartphone, in the cloud, or hybrid, the user (holder) can send the user credentials to the service provider (verifier) in order to access a service. As the user is in control of their data, this paradigm may help to promote privacy. The verifier can verify that the credentials were actually issued by an identity provider (issuer) and sent by the holder using cryptographic signatures. These verifiable credentials (VCs) are a collection of claims that make statements about the holder, such as an email address and name. Information about all these entities is typically stored decentralized, utilizing distributed ledger technologies (DLTs), such as blockchain. However, other—already practically explored—methods could be used instead.

For eIDAS, the protocols and type of storage still remain unclear. This means that DLT or other technologies (e. g., public key infrastructure (PKI) or European Telecommunications Standards Institute (ETSI) trust list) could be used in conjunction with the protocols of OpenID Connect (OIDC) [5], Self Issued OpenID Provider (SIOP) [6], OpenID for Verifiable Presentations (OpenID4VP) [7], and OpenID Federation [8] and various standards for SSI. Either way, interoperability with existing infrastructure and security should be provided, as outlined by Schwalm [9]. Although blockchains have been in use for years (see Bitcoin [10]), the application of SSI is relatively new and still evolving. Also, the security threats are not fully explored. Despite the urgent need for security, we found almost no previous literature on the threats to SSI.

In order to provide a first step towards secure SSI, we briefly analyze current SSI wallets on the market and review the scientific literature. Here, we do not solely concentrate on SSI but take literature from neighboring areas (mainly cryptocurrencies) into account. We then evaluate their applicability to the current SSI systems and point out future work. The contribution is multifold: (1) a brief security analysis of current wallets; (2) a literature survey on threats related to SSI and their elements; and (3) an evaluation of the applicability and outline of future work.

The remainder of the article is as follows: We first provide a brief background on SSI and define the terminology used in this article. Next, in Section 3, we motivate the survey with a security analysis of selected SSI wallets found in the Google Play Store. In Section 4, we outline our methodology, before we provide an overview of the literature survey in Section 5. This is followed by an in-depth summary of the vulnerabilities, threats, and attacks in Section 6 (blockchain applications) and Section 7 (SSI). Since we included related work outside SSI, we evaluate the applicability in Section 8. Additionally, we identify gaps, which lead to future work. Finally, we conclude the article.

2. Background on SSI

Allen [11] first proposed the ideas and principles that form the current design goals of SSI. The ten principles can be summarized as existence, control, access, transparency, persistence, portability, interoperability, consent, minimization, and protection. According to Allen, by implementing selective disclosure, developers can facilitate data minimization better and thereby support privacy. Due to putting the user at the center of any authentication, the users must protect their identities to avoid data breaches. Besides confidentiality and integrity, this also includes the availability of users' data and any associated claims.

The following sections introduce the terminology used to describe SSI systems and components (see Section 2.1), provide an overview of common SSI architectures (see Section 2.2), and showcase relevant implementations of SSI systems (see Section 2.3). As the literature survey includes publications from other areas, a general outline of blockchain and DLT is provided (see Section 2.4).

2.1. Components & Terminology

Put together, SSI implements Allen's design goals by having the user—usually combining the roles of holder and subject—in the center of all transactions and in control of their data through a digital wallet. The wallet can be on a smartphone, in the cloud, or both. Holders receive verifiable credentials describing specific claims from issuers, which are usually organizations, such as government agencies, universities, insurance companies, and banks. The user can then show the verifiable credentials as proof to service providers—acting as verifiers—to gain access to their services. Agents and hubs can be used to provide users with persistent and always-online endpoints. Information about the participating entities, such as decentralized identifiers (DIDs), is stored in a decentralized manner. This could be a DLT, such as a blockchain, or any other decentralized form of storage. Some SSI solutions may also integrate smart contracts to automate processes on the DLT.

For the system described above, we apply the terminology based on Preukschat et al. [12] and Mühle et al. [13]:

- Verifiable Credentials: A collection of metadata and claims about an entity that can be verified by a proofing mechanism.
- Claim: Statement about an attribute of an entity, such as age or email address.
- Proof: Data (digital signature) allowing a verifiable credential to be verified by a verifier.
- Wallet: Software to store private keys, verifiable credentials, and other documents of an entity.
- Verifier: Allows access to a service after receiving the requested information or attributes of a holder.
- Issuer: Trusted parties that verify attributes or claims of an entity.

- Subject: The entity the claims within the verifiable credential are made about.
- Holder: Owner of the claims within a verifiable credential, stored in a wallet, and usually the same entity as the subject.
- Agent and hub: Technical endpoints and trustees for identifiers, which enable communication between entities.
- Smart contract: A computer program or transaction protocol that is executed automatically according to the terms of a contract or an agreement.

2.2. SSI Architecture

Reminiscent of the layered Open Systems Interconnection (OSI) model or transmission control protocol (TCP)/Internet protocol (IP) architecture, the components of an SSI system are organized in a similarly layered architecture. Most often, the architecture described by the Hyperledger Aries project [14] is used. This architecture consists of four layers, as shown in Figure 1. The first, bottom layer defines the technology used to store and express credentials. In the case of the Hyperledger Aries project, this layer is implemented on the Sovrin Ledger [14], but, in general, a variety of blockchains, distributed ledgers, or other decentralized databases can be used. This layer is necessary for specifying identifiers (DIDs) and associating keys to them. On top of that, the second layer adds communication protocols to securely exchange messages between agents, wallets, and hubs.

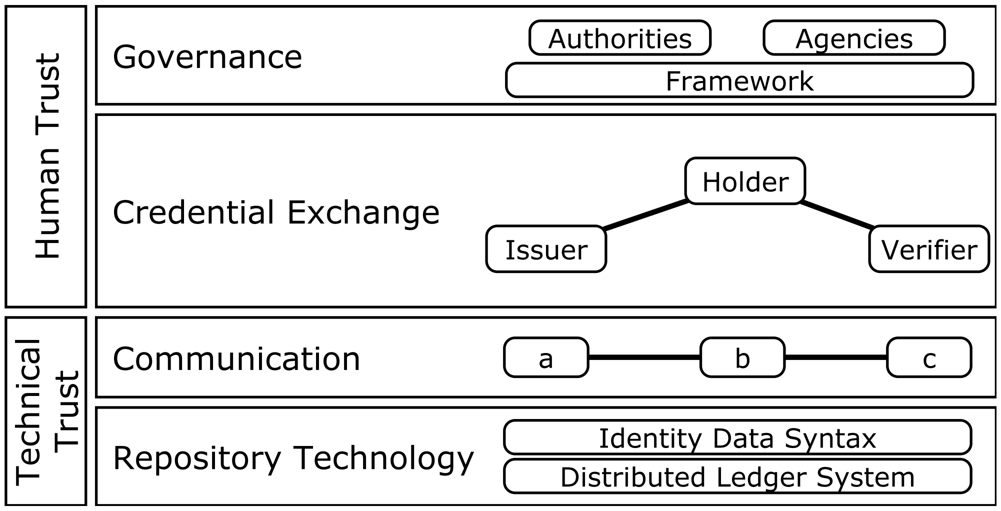


Figure 1. Typical layers of SSI architectures adapted from [14].

Providing the technical premises, layers one and two are used as the foundation to build organizational and human trust at layers three and four. Layer three defines the process of creating, exchanging, and presenting verifiable credentials. Allowing those credentials to be used in more than one specific setting is the goal of layer four. It spans a governance structure, helping organizations to cooperate and accept each other's credentials.

2.3. Current Examples

The SSI wallets selected for closer inspection in Section 3.1 support different implementations of SSI. The most common implementation is based on the SSI protocol, initially developed by Sovrin and continued as Hyperledger Indy and Aries by the Linux Foundation [14]. Hyperledger Indy contains an implementation of a permissioned distributed ledger that is usually run in a consortium. This represents technology on layer one of the architecture described in Section 2.2. This base layer is enhanced by Hyperledger Aries, which provides tools for generating and communicating verifiable credentials. It can be used to build applications for architecture layers two and three.

2.4. Blockchain & DLT

Blockchain technology is closely related to SSI. Section 5 surveys the security issues of blockchains and DLT. In general, blockchains are a subset of DLT. They are characterized by combining transactions into blocks, which are committed to the ledger. DLT in general can use any number of methods to implement a distributed, append only data storage system.

Common to all DLT is the need to form the consensus among all participants about the current state of the data store. This consensus protocol is one of the main differences between blockchain implementations. Bitcoin famously uses a proof-of-work (PoW) approach, where miners, i. e., the participants aiming to combine pending transactions together to write an updated state, need to solve a cryptographic puzzle in the form of calculating a hash value with specific properties [15]. The hash values can only be generated by brute-forcing inputs until the output fulfills the needed properties. Additionally, properties can be adjusted to keep the puzzle hard or easy enough, depending on the available processing power.

As PoW is criticized for its "useless" waste of processing power, many DLT projects are looking for suitable replacements. Ethereum uses a proof-of-stake (PoS) approach, which does not need the raw processing power and is seen as a replacement for PoW [16]. Non-public DLT can use a simpler form of consensus, where the duty to update the database's state is deterministically decided.

Besides storing transactions, some DLTs also feature so-called smart contracts, which are programs that are "run on the DLT" [17]. Like a transaction, every participant can view the smart contract's code and each invocation's inputs to verify the output, which is computed by the miner as part of updating the state.

3. Motivation for the SSI Wallet Survey

With SSI, the user receives self-sovereign control over their data. This data is typically stored in a smartphone wallet. Hence, their usability, functionality, and security are important. We first outline our methodology and then present the results.

3.1. Methodology for Evaluating SSI Wallets

To evaluate real-world SSI wallets, we used the wallets listed by the European Blockchain Association [18]. Wallets found in the Google Play Store are installed and tested on a smartphone Pixel 6 with the current Android operating system (OS) version 13 at the time of the study. Thereby, we obtained Lissi Wallet [19], Verimi [20], Data Wallet by iGrant.io [21], esatus Wallet [22], VIDwallet [23], SmartWallet by Jolocom [24], and Gataca Identity [25] (see Table 1). These are also available in Apple's App Store. We then evaluate the wallets in a four-step process.

Table 1. Android SSI wallets and versions.

Wallet	Organization	Version	SSI Flavor
DataWallet	iGrant.io	3.4.1	W3C, X.509
esatus Wallet	esatus	1.13	W3C
Gataca Identity	Gataca	1.14.1	W3C
Lissi Wallet	Lissi	1.8.1.	Lissi
SmartWallet	Jolocom	2.6.0	W3C
Verimi	Verimi	2.7.0	Verimi
VIDwallet	ValidatedID	1.7.19	W3C

1. We first install the apps, instantiate individual wallets, and summarize the first impression.
2. Each of the organizations offers at least one demo workflow, which we used to play with the wallets. Here, we focus on usability and functionality.

3. Based on Allen's [11] principle of transparency, the algorithm should be open source. We assume that this is also true for the wallet software. Hence, we search for code repositories, which are likely located on GitHub, and take a look at the source code.
4. Additionally, we analyze the Android packages (APKs) with (1) Android Studio [26], (2) APKHunt [27], and (3) manually with the criteria provided by Uddin et al. [28]. If the wallet is not working properly on the smartphone, we additionally evaluate it with a virtual device in Android Studio [29]. If the app requires secure elements, this step may fail.

Background Android applications are distributed as APK files, which are basically ZIP files similar to the Java archive (JAR) files used to package Java libraries. An APK file contains the app code in Dalvic executable format (DEX), native libraries, resources, assets, and an Android manifest in binary extensible markup language (XML). The APKs are typically signed.

Limitations We note suspicious behavior while using the wallets and analyze the wallets statically with the help of tools and specific terms. Hence, we present the early results of the security analysis of the wallets. A more thorough analysis could include more terms and dynamic analysis, among others. In addition, our analysis was conducted with the versions noted. We did not verify that the issues exist in versions published after the analysis. Lastly, we did not consider apps in the Apple App Store, such as Apple Wallet or Verimi ID-Wallet for iOS.

3.2. Results of the Evaluation

In the following, we present our results step by step.

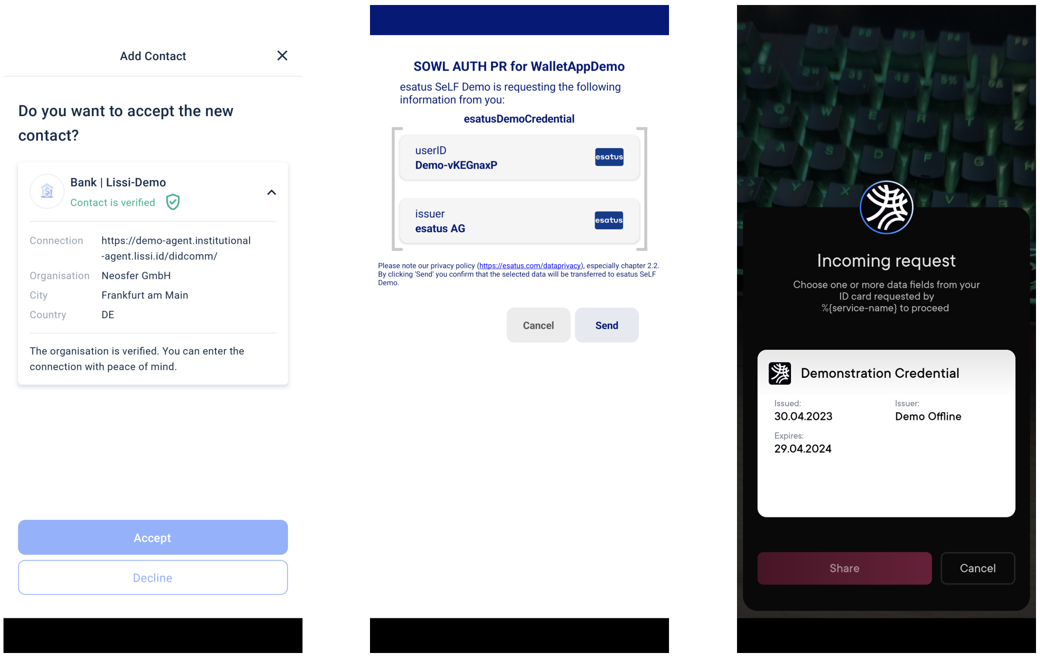
3.2.1. Installation, Instantiation, and First Impression

We installed all apps and managed to instantiate individual wallets on all except two: Verimi Wallet and Gataca Identity. We observed similar design elements and workflows throughout the wallets. For example, scanning a new quick response (QR) code to receive or send credentials is possible via a button with the corresponding function mostly in the middle of the bar menu. The user can go through the received VCs by swiping or clicking. Almost all apps use a personal identification number (PIN) with four to six numbers for the first authentication; biometric authentication is typically available afterward. The exception is the Data Wallet, which does not require any authentication.

3.2.2. Wallet Usage

When receiving claims, the user can either accept or decline the attempt. In one case (VIDwallet), QR codes cannot be scanned twice. If one workflow does not work, then this will stop the user from proceeding. We shortly summarized the findings during wallet usage in [30]. Few wallets allow self-asserted claims (VIDwallet and SmartWallet). However, the requests may fail nonetheless in the case of the VIDwallet. The amount of information on the issuer and verifier varies from wallet to wallet. Some only display a logo and basic information (esatus Wallet and SmartWallet), whereas others provide a long list (Data Wallet) or visual signs that the entity is verified (Lissi Wallet). The three examples of the Lissi Wallet (see Figure 2a), the esatus Wallet (see Figure 2b), and the SmartWallet (see Figure 2c) are shown in Figure 2.

In the case of the Data Wallet, the data on claims is blurred similar to passwords. This can enhance the perspective that the data is as important as passwords. However, this could also lead to sharing data without thinking. The difference between the Data Wallet (see Figure 3b,c) and another wallet, the Lissi Wallet (see Figure 3a), is shown in Figure 3. In the Data Wallet, users can unblur the claims by clicking on the eye button. However, this is not mandatory. On the contrary, most wallets, such as Lissi Wallet, show all information in clear text, which can also be critical.



Add Contact

Do you want to accept the new contact?

Bank | Lissi-Demo
Contact is verified

Connection	https://demo-agent.institutional-agent.lissi.id/didcomm/
Organisation	Neosfer GmbH
City	Frankfurt am Main
Country	DE

The organisation is verified. You can enter the connection with peace of mind.

SOWL AUTH PR for WalletAppDemo

esatus SeLF Demo is requesting the following information from you:

esatusDemoCredential

userID	Demo-vKEGnaxP
issuer	esatus AG

Please note our privacy policy (https://esatus.com/dataprivacy), especially chapter 2.2. By clicking 'Send' you confirm that the selected data will be transferred to esatus SeLF Demo.

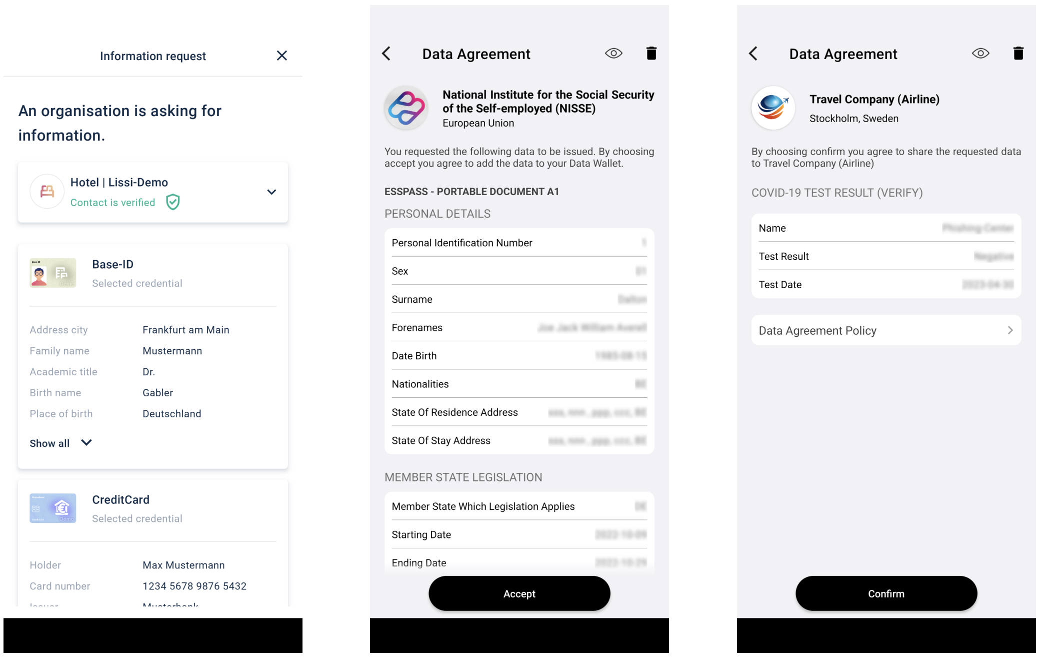
Incoming request

Choose one or more data fields from your ID card requested by %{service-name} to proceed

Demonstration Credential

Issued: 30.04.2023	Issuer: Demo Offline
Expires: 29.04.2024	

Figure 2. Screenshots of selected wallets found in the Google Play Store 1/3. (**a**) Lissi Wallet receiving a request. (**b**) esatus Wallet receiving an information request [30]. (**c**) SmartWallet with an incoming request [30].



Information request

An organisation is asking for information.

Hotel | Lissi-Demo
Contact is verified

Base-ID
Selected credential

Address city	Frankfurt am Main
Family name	Mustermann
Academic title	Dr.
Birth name	Gabler
Place of birth	Deutschland

CreditCard
Selected credential

Holder	Max Mustermann
Card number	1234 5678 9876 5432

Data Agreement

National Institute for the Social Security of the Self-employed (NISSE)
European Union

You requested the following data to be issued. By choosing accept you agree to add the data to your Data Wallet.

ESSPASS - PORTABLE DOCUMENT A1

PERSONAL DETAILS

Personal Identification Number
Sex
Surname
Forenames
Date Birth
Nationalities
State Of Residence Address
State Of Stay Address

MEMBER STATE LEGISLATION

Member State Which Legislation Applies
Starting Date
Ending Date

Data Agreement

Travel Company (Airline)
Stockholm, Sweden

By choosing confirm you agree to share the requested data to Travel Company (Airline)

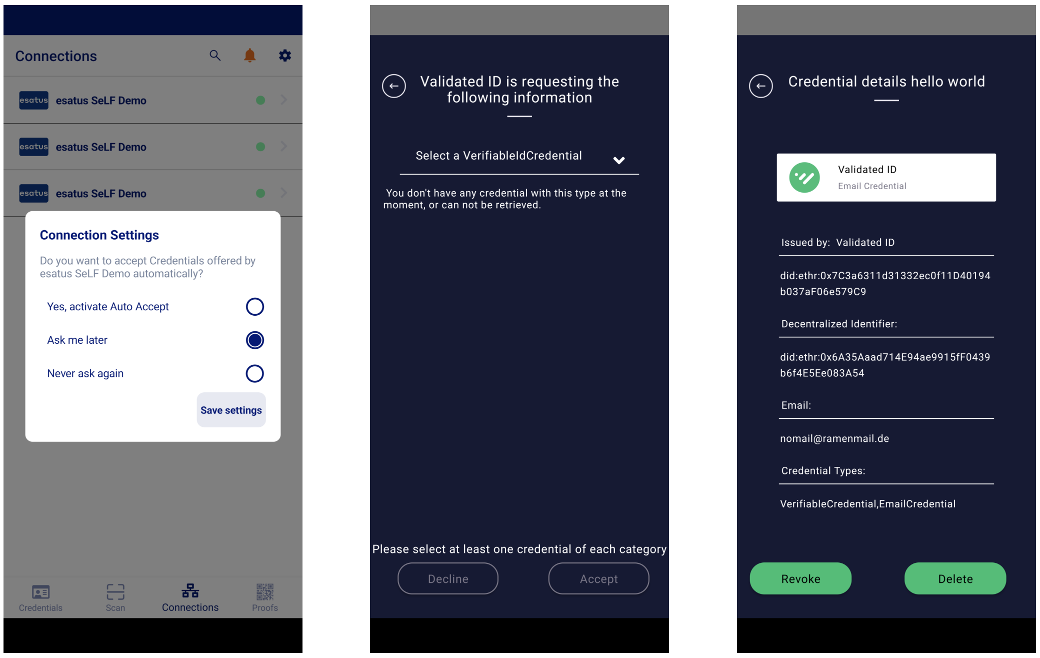
COVID-19 TEST RESULT (VERIFY)

Name
Test Result
Test Date

Data Agreement Policy

Figure 3. Screenshots of selected wallets found in the Google Play Store 2/3. (**a**) Lissi Wallet receiving an information request [30]. (**b**) iGrant.io Data Wallet receiving blurred claims [30]. (**c**) iGrant.io Data Wallet receiving a blurred information request.

In the case of the esatus Wallet, the issuer or verifier can be set to automatically accept (see Figure 4a). On the one hand, this may reduce the number of notifications for the user, so they can pay attention to those that appear. On the other hand, the entity may change its behavior to a malicious one without the user being notified. The default setting is being asked next time. However, if the user decides not to be asked again, then the default setting is to receive no notifications about interactions. Default notifications may provide a higher degree of transparency. In addition, we missed information about redirects and other actions in the VIDwallet (see Figure 4b). The VIDwallet is one of the few wallets that accept self-issued credentials. However, it is debatable how qualitative they are, as shown in Figure 4c. Here, we used a throwaway email address to receive a verifiable email address credential.


Connections

esatus SeLF Demo

esatus SeLF Demo

esatus SeLF Demo

Connection Settings

Do you want to accept Credentials offered by esatus SeLF Demo automatically?

Yes, activate Auto Accept

[✓ Ask me later]

Never ask again

Save settings

Validated ID is requesting the following information

Select a VerifiableIdCredential

You don't have any credential with this type at the moment, or can not be retrieved.

Please select at least one credential of each category

Credential details hello world

Validated ID
Email Credential

Issued by: Validated ID

did:ethr:0x7C3a6311d31332ec0f11D40194b037aF06e579C9

Decentralized Identifier:

did:ethr:0x6A35Aaad714E94ae9915fF0439b6f4E5Ee083A54

Email:

nomail@ramenmail.de

Credential Types:

VerifiableCredential,EmailCredential

(**a**) (**b**) (**c**)

Figure 4. Screenshots of selected wallets found in the Google Play Store 3/3. (**a**) esatus Wallet asking about future behavior with verifier [30]. (**b**) VIDwallet with issues complying with requested claims [30]. (**c**) VIDwallet receiving claims about a throwaway email address.

Based on these observations, we could not find guidance for the user to decide whether to trust and accept the offer or request of an entity. For example, we only noticed information about issuers and verifiers being certified once. Also, we could not see if the amount of data requested was typical or more than required. Both design variants of Data Wallet and esatus Wallet can either help reduce the burden or lead to just accepting everything. The issue is that too many notifications (see cookie banners or warnings) may make the users tired, whereas too little information may lead to a similar result. With SSI, the user has more possibilities but also more responsibilities. Consequently, further work is required to find suitable designs to guide users.

3.3. Further Usage and Analysis

We observed that several wallets disabled Android backups, allowed clear-text traffic, and used older methods such as the message-digest algorithm 5 (MD5), secure hash algorithm 1 (SHA1), and data encryption standard (DES) and older secure sockets layer (SSL)/transport layer security (TLS) versions or applied custom crypto methods. In addition, we were able to gain access to the SQLite database of Data Wallet and find Alice and Bob in the log files. For esatus Wallet, we noticed a connection to Azure. However, we were not sure what, for example, the "Fitness Activity" is used for. In one case (Gataca Identity), we observed debugging functionality and user certificates were allowed. More interestingly, in the case of VIDwallet, we found a Facebook token and key in the source code and our pictures taken of IDs on the file system.

Uddin et al. [28] describe specific application programming interface (API) calls, which indicate feature inclusion.

- Root detection: The app checks if it can execute the `su` command or if any root-enabled apps are on the smartphone.
- Integrity check: It verifies that the app has not been tampered with and was installed from a verified app store.
- Custom keyboard: Since keyboards store inputs independently whether they are sensitive or not, passwords and similar information from a wallet should not be included. One way of mitigating this is by using custom keyboards.
- Biometric authentication: The use of biometrics or two-factor authentication (2FA) as a type of multi-factor authentication (MFA).
- Screenshot disabled: Malicious apps and users can make screenshots of credentials, passwords, and similar. Hence, disabling this feature can help.
- Hardware secure module: The key-revealing information is stored in a secure enclave to provide a higher level of security.
- Random generator: A secure random generator used for cryptography.

The authors specified those for crypto wallets. However, these indicators should also apply to SSI wallets. We evaluated the selected SSI wallets based on these API calls. Most do not include these features besides secure random generation and biometric authentication, as shown in Table 2.

Table 2. Assessment of the selected wallets based on the criteria in [28].

Type	API Signature	DW	eW	GÍ	SW	LW	V	VID
Root Detection	Runtime.exec()	-	-	-	-	-	-	lib
	PackageManager.getPackageInfo()	-	-	-	-	-	-	-
	Os.stat()	-	-	-	-	-	-	-
	Os.access()	-	-	-	-	-	-	-
Integrity Check	PackageManager.getPackageInfo()	-	-	-	-	-	-	-
	Context.getPackageCodePath()	-	-	-	-	-	-	-
	ZipFile.init()	-	-	-	-	-	-	-
	RandomAccessFile.init()	-	-	-	-	-	-	-
Custom Keyboard	KeyboardView.setKeyboard()	-	-	-	-	-	-	-
	OnKeyboardActionListener.onKey()	-	-	-	-	-	-	-
	InputMethodService.onCreateInputView()	-	-	-	-	-	-	-
	InputConnection.commitText()	-	-	-	-	-	-	-
	InputMethod	+	+	+	+	+	+	+
Biometric Authentication	BiometricManager	+	+	+	+	+	+	+
	BiometricPrompt	+	+	+	+	+	+	+
	FingerprintManager	+	+	+	+	+	+	+
	BiometricService	-	-	-	-	-	-	-
	FingerprintService	-	-	-	-	-	-	-
Screenshots Disabled	Windows.setFlags()	-	-	-	-	-	-	-
	View.setDrawingCacheEnabled()	-	-	-	-	-	-	-
Hardware Security Module	KeyStore.getInstance()	-	-	-	-	-	-	-
	KeyGenParameterSpec.Builder.isStrongBoxBacked()	-	-	-	-	-	-	-
	StrongBoxUnavailableException	-	lib	+	-	-	+	-
Random Generator	SecureRandom	+	lib	+	+	+	+	+

Abbreviations: DW: Data Wallet; eW: esatus Wallet; GÍ: Gataca Identity; SW: SmartWallet; LW: Lissi Wallet; V: Verimi; VID: VIDwallet; lib: provided in a library.

3.4. Summary

Although the apps are officially released in the Play Store, we noticed some issues during the demo workflows, such as not being able to scan QR codes (SmartWallet in case of bigger codes), having problems loading VCs (Lissi), and invalid QR codes (VIDwallet). Based on these issues and the updates made to the GitHub repositories, we assume that by the time of writing this article, the wallets are in a premature stage. When productively using wallets, these should be securely designed, use current versions of libraries and functionalities, and do not include any kind of trackers.

4. Methodology of the Survey

Following the methodology described by Page et al. [31], we conducted a systematic literature review. Publications related to the threats of SSI are collected by applying specific search terms to pre-defined publishers. The research questions filter the publications due to exclusion criteria to assess the quality and focus of the papers' contents. Each of the stages is described below. The applied methodology is visualized in Figure 5.

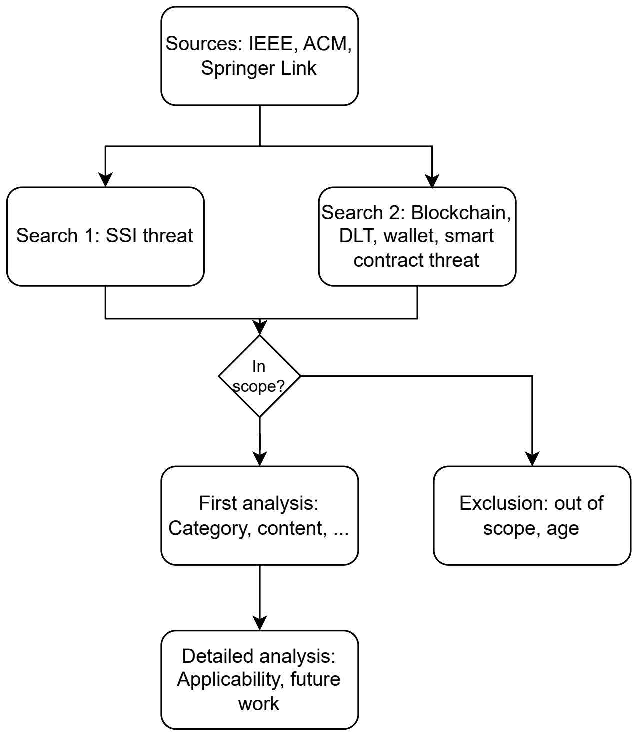

Sources: IEEE, ACM, Springer Link
Search 1: SSI threat
Search 2: Blockchain, DLT, wallet, smart contract threat
In scope?
First analysis: Category, content, ...
Exclusion: out of scope, age
Detailed analysis: Applicability, future work

Figure 5. The methodology applied in the literature survey.

4.1. Research Questions

This article aims to systematically review threats to SSI and its elements. In particular, we review the threats to blockchain/DLT, smart contracts, and wallets and intend to present a detailed perspective on their application to SSI. By comparing the current SSI technologies and the applicability of the threats, we intend to identify gaps, leading to future research. Thereby, we receive the following research questions.

RQ1: Which current threats are known for SSI and their elements? This research question aims to provide a full picture of possible threats known in the literature.

RQ2: Which are the known countermeasures for these threats? This research question tries to identify known countermeasures.

RQ3: Which of these identified threats and countermeasures are actually applicable to SSI? Since we regard various forms of these elements, some might not be relevant for SSI. Therefore, this research question focuses on those threats and countermeasures that are applicable.

RQ4: Which parts of SSI have not been regarded by literature and which are the next steps? This research question aims to identify gaps, leading to future work.

RQ1 and RQ2 focus on the literature survey (Section 6 for blockchain applications and Section 7 for SSI), whereas RQ3 and RQ4 are relevant for the following evaluation and discussion (Section 8).

4.2. Exclusion Criteria

We exclude the following publications.

EC1: Publications before 2020 (publication date), since the threats have to be able to apply for SSI and not for earlier stages of SSI or its elements.
EC2: Posters and short papers, as they typically present preliminary work.
EC3: Publications in other languages than English.
EC4: Publications, which propose their own approach but do not contribute to the threat overview.

Following EC1, we do not follow the original publications. In addition, we remove duplicates.

4.3. Search Terms and Process

The searches were conducted in June 2023. In order to search for literature concerning the threats of SSI or elements of it, we first search (Search 1) specifically for papers on SSI. We use the search string `‘‘self-sovereign identity’’ AND (security OR attack OR threat)` as the basis for ACM, IEEE, and Springer Link. We adapt the search terms and filters according to the publisher, as summarized in Table 3.

Table 3. Search terms for literature on SSI threats.

Publisher	Search String	Results	Relevant
ACM	[Abstract: "self-sovereign identity"] AND [[Abstract: security] OR [Abstract: attack] OR [Abstract: threat]]	3	0
IEEE	("Abstract":"self-sovereign identity") AND (("Abstract":security) OR ("Abstract":attack) OR ("Abstract":threat))	56	2
Springer Link	Basis with limitation to language = English, discipline = Computer Science, and Document publication = 2020–2023; in addition without books	69	1

As we find almost no relevant publications but want to provide a full picture, we additionally search (Search 2) for threats to elements that might be used for SSI. These are blockchain, DLT, smart contracts, and wallets. Hence, we apply the search string `(wallets OR blockchain OR dlt OR ‘‘smart contract’’) AND (security OR attack OR threat)` as a basis and adapt it for ACM, IEEE, and Springer. The result is summarized in Table 4.

Table 4. Search terms for literature on threats for SSI elements.

Publisher	Search String	Results	Relevant
ACM	[[Abstract: security] OR [Abstract: attack] OR [Abstract: threat]] AND [[Abstract: wallet] OR [Abstract: blockchain] OR [Abstract: dlt] OR [Abstract: "smart contract"]] AND [E-Publication Date: (01/01/2020 TO 12/06/2023)]	830	36
IEEE	(("Abstract":security) OR ("Abstract":threat) OR ("Abstract":vulnerability)) AND (("Abstract":wallet) OR ("Abstract":dlt) OR ("Abstract":blockchain) OR ("Abstract":"smart contract")) and restriction to 2020–2023, publication type (Conferences, journals, early access articles, magazines), publisher (IEEE), and topics (blockchains, contracts, cryptocurrencies)	3356	37
Springer Link	(wallet OR blockchain OR dlt OR "smart contract) AND (security OR attack OR threat)	600	27

4.4. Analysis

Next, we analyze the contents for relevance and group the remaining papers into the categories of wallet, DLT and blockchain, smart contracts, and humans (first analysis). During this step, we summarize all known threats, vulnerabilities, and security issues. In the final step (detailed analysis), we evaluate the applicability of the threats to SSI. This has several reasons: not all publications found focus on SSI and the applied technologies may differ (such as applications and consensus algorithms). Based on the results, we identify missing pieces, leading to future work.

4.5. Limitations

The literature survey concentrates on the publishers of IEEE, ACM, and Springer Link. It excludes others, where further literature could be published, and cannot be complete. As we choose major publishers, the most known publications should be included. The categorization is based on the title and the content of the paper. Depending on the actual paper, other categories might be involved. We chose the most prominent one, appearing in the title. Additionally, we exclude publications before 2020, since the field is rapidly evolving. Lastly, we focus on threats and vulnerabilities in this survey. We note countermeasures and summarize them per layer, but these might not be all the measures that can be taken.

5. Overview of the Literature Survey

Based on the search with the search strategy described in the section above, we received several results for the categories of wallets, DLT/blockchain, smart contracts, and humans. The number of publications and their references are summarized in Table 5.

Table 5. Literature grouped into categories.

Category	Publications	No. of Publications
SSI	[32–34]	3
Wallet	[28,35–43]	10
DLT/Blockchain	[44–82]	39
Smart Contracts	[83–128]	46
Humans	[129–133]	5

When regarding the year of publication, we notice the following distribution, as shown in Table 6. Although SSI is gaining momentum, the number of publications focusing on threats is constant. Wallets and humans were considered at the beginning of the search period. Also, the number of publications regarding smart contracts and DLT seems to be decreasing. Since we conducted the search in June, the number of publications in 2023 may increase until the end of the year. Thereby, it is difficult to see any tendency.

Table 6. Literature grouped into categories and years.

Year	SSI	Wallet	DLT	Humans	Smart Contract
2020	0	3	10	2	16
2021	1	6	13	3	15
2022	1	0	13	0	13
2023	1	1	3	0	2
all	3	10	39	5	47

Most of the included publications have no or less than ten distinct citations; 15 publications have 10–50 citations, six publications have 51–100 citations, and three have more than 100 citations. All three provide an overview of threats or security for their area (blockchain, smart contracts, and Ethereum).

We additionally review the categories provided by the authors, which are included in the BibTeX files for ACM and Springer Link. We never find the category threats, but threat model (1), security and derivations of it (16), vulnerability (2), attacks (1), and attack detection (1). Regarding our categories, most publications state blockchain (18) and smart contract (14) or derivations. Further categories include wallets (3), distributed ledger technologies (2), consensus (1), and consensus algorithms, such as proof-of-work or proof-of-authority (2). Regarding the use case, we notice cryptocurrencies (7), Bitcoin (5), Ethereum (6), and Monero (1). Specific attacks or threats are less often stated: software supply chain (1), application security (1), deep fake (1), fork after withholding (FAW) attack (1), oracle manipulator (1), reentrancy vulnerability (1), routing attack (1), selfholding attack (1), selfish mining (1), social engineering (2), and timestamp attack (1).

Based on the content, several authors mention theft, malware, and spoofing in general. Regarding vulnerabilities, we most often find the reentrancy vulnerability, gas-related issues, issues with Ether, short addresses, unprotected suicide, integer overflow or underflow, erroneous visibility, authentication through `tx.origin` (tx stands for transaction), timestamp dependency, selfish mining, double-spending, and transaction ordering dependence. The Parity multi-signature wallet attack, the 51% attack, the Sybil attack, and the Eclipse attack are the most prominent attacks. The majority of publications do not group these vulnerabilities, issues, and attacks based on the blockchain layers, but describe them one by one. These and other threats are outlined in the next section.

6. Results of the Blockchain Survey

In the following, we describe the result of our survey by the layers, established in the area of blockchain: human layer (see Section 6.2), application layer (see Section 6.3), consensus layer (see Section 6.4), data layer (see Section 6.5), and network layer (see Section 6.6). A more detailed overview can be found in Table 7.

Table 7. Overview of the blockchain survey with layers, categories, and issues.

Layer	Category	Threat/Vulnerability/Attack	Section
General		DoS, DDoS, and more	Section 6.1
Human	Social engineering	Phishing, Identity theft, shoulder surfing	Section 6.2.1
	Human errors	Human errors, deanonymization	Section 6.2.2
	Wallet	Physical threats, keyboard, backup, malware, spoofing	Section 6.2.3
Application	Inter-contractual	Reentrancy vulnerability, gas-related issues	Section 6.3.1
	Contractual	Issues with Ether, upgradeable contract and backdoor, honeypot, address issues, unprotected suicide, DoS with unexpected revert, integer overflow/underflow, confidentiality failure, insufficient signature information	Section 6.3.2
	Contract-programming	Specific issues, delegatecall injection, erroneous visibility, authentication through tx.origin, manipulated balance, unchecked call return values, uninitialized storage pointer, call to unknown, type casts, outdated compiler version, permission control	Section 6.3.3
	Transaction	Call-stack depth limit, timestamp dependence	Section 6.3.4
Consensus	Mining	Honest mining assumption, misleading rewards, probabilistic finality, transaction vulnerability, verifier's dilemma	Section 6.4.1
	Timing	Timing vulnerabilities, transaction order	Section 6.4.2
	Others	Strength of algorithm, spoofing, collusion attack, and more	Section 6.4.3
Data		Indistinguishable chains, empty account in the state trie, trusted third parties	Section 6.5
Network	Blockchain-specific	Forgery attack, unlimited node creation, uncapped incoming connections, public/fixed peer selection, sole block synchronization	Section 6.6.1
	Network	Impersonation attack, replay attack, Sybil attack, Eclipse attack, API exposure	Section 6.6.2

Most publications focus on the cryptocurrency Ethereum. Since these may not be applicable to SSI, we discuss the findings in Section 8. As the results of the literature survey on SSI differ from those of the blockchain search in handled data, applications, consensus algorithms, and involved parties, we briefly summarize these findings in the following section (see Section 7).

6.1. General Issues

A few publications did not fit into any layers. Huang et al. [64] include security in their survey on blockchains without going into detail. Sayeed et al. [116] describe malicious acts (for example, phishing), weak protocols, defrauding (i. e., fraudulent users), and application bugs (for example, code errors), which can happen at each layer. Raikwar and Gligoroski [46] outline various types of denial-of-service (DoS) or distributed DoS (DDoS) attacks, i. e., on wallets, on cryptocurrency exchange services, on memory/transaction pools, on mining pools, on layer-two blockchain protocols, on sharding protocols, which is underlined by Li et al. [61], on commit-chain operators, on smart contracts, on mixing services, and on consensus participants. The authors describe the mitigation techniques of client puzzles, fee-based approaches, and DoS-resistant protocols. Similarly, Mirkin et al. [62] outline DoS in PoW, naming it Blockchain DoS (BDoS). Hu et al. [38] analyze the amplification spamming attack applied for DoS. This shows that there are several ways for DDoS attacks.

6.2. Human Layer

Since some publications include human aspects, such as social engineering, we add the human layer to the blockchain layer model. Whereas most publications concentrate on the end-user, Fröhlich et al. [131] explicitly state staff members. In the following, we summarize the results, grouped into the categories of social engineering (see Section 6.2.1), human errors and accidental threats (see Section 6.2.2), and wallet threats (see Section 6.2.3), which can have various reasons. Lastly, we state some countermeasures in Section 6.2.4.

6.2.1. Social Engineering

Fröhlich et al. [132] describe attacks on the owner, forcing them to provide access to the wallet, and betrayal. However, more variants of social engineering are applicable.

Phishing According to Wilusz and Wójtowicz [35], phishing is a form of social engineering where attackers deceive victims into revealing sensitive information or installing malware. In the field of blockchain, the variants of asking for the private key, credentials, or cryptocurrency (as exchange or contribution) exist, according to Weber et al. [129]. Fröhlich et al. [131] name email phishing, ad phishing, social media phishing, voice phishing, short message service (SMS) phishing, and spear-phishing. In addition, the authors state the variants of exit scams, fraudulent cryptocurrency scams, transaction scams, impersonation giveaway scams, and blackmail scams. Related to scams, Weber et al. [129] describe honeypot scams as publishing the private key to a wallet containing several tokens that do not exist.

Identity theft: Identity theft describes the usage of another's personal identifying information without their permission, according to Fröhlich et al. [131]. This could happen in blockchain use cases after phishing, but also following data breaches and other attacks. Wilusz and Wójtowicz [35] outline password guessing and biometric side-passing. Do et al. [36] apply deep fakes against liveliness detection during biometric authentication. Attacks on third-party services, such as subscriber identity module (SIM) swapping attacks, can have implications for the usage of wallets [35,131].

Shoulder surfing: Shoulder surfing describes the act of watching the victim type and using this information for ill-purposes. Swambo and Poinsot [83] outline the compromise of the key backup by observing. However, further variants, such as watching the authentication on the smartphone or wallet, are possible.

6.2.2. Human Errors and Accidental Threats

Human errors and accidental threats can lead to deanonymization, among others.

Human errors: Human errors [35,131,132] are often stated in the analyzed literature. This includes erroneous recording of access credentials, loss of access credentials (forgetting, destruction, equipment breakdown, and destructive catastrophes), and erroneous transactions (misspelled address, amount, or fees) [125,131]. Issues with addresses are also stated by Ferreira Torres et al. [101] and Ivanov et al. [130].

Deanonymization: Deanonymization is generally named by Fröhlich et al. [131] and others. Romiti et al. [85] explain deanonymization by heuristics and linking algorithms, whereas Apostolaki et al. [86] use perimeter by traffic interception. Ghesmati et al. [87] describe the following deanonymization techniques that can be applied to the cryptocurrency Bitcoin: heuristics, side-channel attacks, flow analysis, and auxiliary information. Within heuristics, the authors state input ownership, change address, address reuse, single input single output, cluster growth, and specific patterns. For side-channel attacks, they explain time correlation, amount correlation, network layer information, and cashing out on forks. Flow analysis is aided by transaction graphs, taint analysis, and user graphs. Auxiliary information includes forums, websites, search engines, social networks, service APIs, interactions, an address tag database, and others.

6.2.3. Wallet Threats

Various wallet threats can be seen. As several of these threats can be caused by human errors, they are described in the human layer.

Physical threats: Physical threats to the wallet and the smartphone include losing and theft (credentials and backups) [35,97,132,133]. Additionally, Fröhlich et al. [131] name the variants of vandalism, extortion, and abduction. He et al. [42] and Li et al. [41] outline the misuse of the debugging mode, for example, for accessibility features.

Keyboard: Uddin et al. [28] analyze the security of cryptocurrency wallets. The keyboard app uses a user dictionary, which is a database of words, locales, and frequency counts, for predictive text inputs. This dictionary can be abused to predict mnemonic phrases by extracting frequency information from typed words if the attacker has access to it. However, multiple apps can have virtual keyboard permissions on a smartphone, as explained by He et al. [41]. Hence, a custom keyboard can prevent this attack.

Backup: If the backup of the app is allowed in the `AndroidManifest.xml` file, the app data is backed up using the Android Debug Bridge (ADB) command, as outlined by Uddin et al. [28]. This enables attackers to retrieve the data. Li et al. [42] describe the impact of unencrypted backup files. Permissions, such as recording audio, can be again misused by attackers. This vulnerability is emphasized by Li et al. [42], explaining the issues for external storage read and write permissions, and He et al. [41] with root privileges. Other verifications and settings may also lead to vulnerabilities, if not done correctly. Similarly, if components are exported, this may lead to leaked information. Both publications also describe the problems with floating windows, which can be misused by malicious apps.

Malware: Generally, malware can be an issue [35,49,133]. Fröhlich et al. [131] differentiate wallet/key extraction malware, transaction manipulation malware, credential extraction malware, and ransomware. Hu et al. [38] and Fröhlich et al. [131] additionally describe the issue of fraudulent client applications, such as the wallet, QR code scanner, or key/wallet generator. Ohm et al. [37] point out the issue of a malicious supply chain, which could be due to a malicious library applied in an application. This threat can be extended to platform risks, as pointed out by Fröhlich et al. [131].

Spoofing: Several authors highlight the private key security, without going into detail [59,74,103]. Hence, proper key management and algorithms are important [41,47,49]. Park et al. [43] and Swambo and Poinsot [83] point out the risk of reconstructing the private key. AlFaw et al. [67] state that digital signatures, hash function, mining malware, software flows, and user address vulnerabilities might be an issue, whereas Dabrowski et al. [40] and Swambo and Poinsot [83] outline firmware, client software, and human-in-the-middle (MITM) of hardware wallets. Spoofing at the user agent layer refers to the illegitimate execution of identity actions, such as disclosing credentials or authorization. The user agent holds the private key and VCs. These could be compromised and modified by the same techniques applied to spoofing.

6.2.4. Countermeasures

Information security awareness and user-centric security are important, according to Weber et al. [129]. Buja et al. [133] propose the countermeasures of good authentication methods, user awareness, up-to-date software, verified or legitimate software, backup, and encryption. Fröhlich et al. [132] recommend redundant backups and awareness, whereas Fröhlich et al. [131] outline being skeptical, accessing platforms directly instead of clicking links, using cold storage, education, and browser extensions warning the user. In addition, the authors suggest comparing transactions, having good user interfaces, recovery methods, redundant backups, and advanced infrastructure for advanced users. Further recommendations by the authors include using trusted sources, 2FA as a form of MFA, a secure passphrase, backups, moving funds to cold wallets, and checking transactions before confirmation. Wilusz and Wójtowicz [35] outline the usage of hardware modules and virtual keyboards. Dabrowski et al. [40] suggest that hardware wallets should be inspected for the packaging, comparing the printed circuit board (PCB) with the reference picture, and verifying the software with hashes or signatures. Agarwal et al. [96] detect phishing, spam, and scams by applying Etherscan, a block explorer and analytics platform for Ethereum. Apostolaki et al. [86] focus on the countermeasures of deanonymization with encrypted traffic, fake peers, obfuscating the client's state, routing-aware transactions' requests and advertisements, and applying Tor or a virtual private network (VPN). Focusing on authentication, the countermeasures of cancelable biometrics, decentralized storage, liveliness detection, device identification, and MFA are suggested by Wilusz and Wójtowicz [35]. Uddin et al. [28] and Naik et al. [42] generally outline detection functionalities, such as regarding permissions and debugging.

6.3. Application Layer

Based on Chen et al. [90], we first provide an overview of vulnerabilities and attacks on the application layer, which primarily consists of smart contracts. We use the categories of inter-contractual (see Section 6.3.1), contractual (see Section 6.3.2), contract-programming (see Section 6.3.3), and transaction (see Section 6.3.4). Lastly, we summarize countermeasures in Section 6.3.5.

Some publications do not clearly differentiate their findings. Bouichou et al. [100] enumerate the issues as privacy and control, storage accessibility, logic, compiler, authentication, cryptography, initiation, wrong attribution of names, arithmetic, useless code, user interface, time constraint, and requirement violation. Sharma and Shak [72] and Snegireva [77] summarize them as faults and vulnerabilities, whereas Hajdu et al. [128] map vulnerabilities to common weakness enumerations (CWEs). These can have different reasons and occur at different places, such as contracts, programming languages, or implementation. Pise and Patil [118] differentiate smart-contract-specific and normal language issues, such as overflows. One example is described by He et al. [95], where an Ethereum explorer mishandled edge cases. As pointed out by Wan et al. [108], 40% of the respondents in their survey reported having experienced at least one out of three potential security problems related to smart contracts.

6.3.1. Inter-Contractual Vulnerabilities

We noticed the following inter-contractual vulnerabilities.

Reentrancy vulnerability: The reentrancy vulnerability [69,76,90,101,102,104,109,114,116,117,119,120,122,125,126] describes a vulnerability, where an external callee contract calls back to a function in the caller contract before the caller contract finishes and, thereby, bypasses the due validity. One example of an attack based on the reentrancy vulnerability is the decentralized autonomous organization (DAO) attack [52,69,125]. The authors further divide the vulnerability into fallback function-based reentrancy and create function-based reentrancy.

Gas-related issues: Several gas-related issues exist [69,117,126]. The king of the Ether allows checking until the caller contract is drained of Ether or the transaction runs out of gas [90,114,117]. Kushwaha et al. [117] further explain DoS with block gas limits or failed calls, gasless sends, gas costly patterns, and insufficient gas. Khan and Siami Namin [126] state DoS, integer overflow, and wallet out of gas. The prevention is implementation-specific (for example, using the transfer function instead of the send function to mitigate the king of the Ether vulnerability).

6.3.2. Contractual Vulnerabilities

Smart contracts may contain vulnerabilities that allow, for example, attackers to gain access to the contract's funds or cause other unexpected behavior, according to Fröhlich et al. [131]. In the following, we describe these vulnerabilities found in the literature.

Issues with Ether: Frozen or locked Ether is the ability of users to deposit their money into their contract accounts with the inability to spend their money from those accounts [90,102,125,126]. This was again a Parity wallet bug. Khan et al. [126] describe stealing Ether, whereas Usman et al. [125] explain unexpected Ether.

Upgradeable contract and backdoor: If the contract developer becomes malicious, the updated contract can get malicious, as outlined by Chen et al. [90]. A (deliberate) backdoor in a smart contract can allow privileged users to withdraw funds. The functionality is typically hidden from immediate detection when inspecting the code (see Fröhlich et al. [131]).

Honeypot: A honeypot contract is a smart contract that pretends to leak its funds to an arbitrary user if the user sends additional funds to it. However, the funds will be trapped so that the attacker can only retrieve them. In addition, the attacker can earn the money the victims send to the contract, as outlined by Fröhlich et al. [131].

Address issues: Leaking Ether to an arbitrary address describes the fact that the contract's funds can be withdrawn by any caller due to failure to check a caller's identity [90,103] or utilizing short addresses [73,90,101,116,119]. This might be applied in combination with social engineering and can (partly) be prevented by enforcing authentication on the functions for sending funds. Similarly, Ether can be lost to orphan addresses [90]. If money is sent to an orphan address, Ethereum automatically registers that address. Hence, the money is effectively lost.

Unprotected suicide: A contract can be killed by the owner using the suicide or self-destruct method [90,103,105,117,120,126]. This vulnerability was first observed during an attack against the Parity wallet and is caused by inadequate authentication enforced by a contract. Following this, it can be mitigated by enforcing adequate authentication methods, meaning that a suicide operation must be approved by multiple parties.

DoS with unexpected revert: The transaction is reverted due to a caller contract encountering a failure in the external call or the callee contract. This deliberately performs the revert operation to disrupt the execution of the caller contract [90,103,126].

Integer overflow or underflow: This vulnerability occurs when the result of an arithmetic operation falls outside of range due to no proper validation [69,101–103,114,116,117,119,120,122,126]. It was first observed during an attack against the BeautyChain (BEC) tokens, as outlined by Chen et al. [90]. The issue is that neither the Solidity compiler nor the Ethereum virtual machine (EVM) provides integer overflow or underflow detection. However, the vulnerability can be prevented by the SafeMath library [134]. Generally, arithmetic bugs, such as unchecked mathematics, can be seen as vulnerabilities.

Confidentiality failure: Restricting the visibility or function does not assure that the variable or function is confidential due to the public nature of blockchain, as explained by Chen et al [90]. The public nature of transactions provides transparency, but also information about the parties involved and their interactions. Hence, privacy breaches can have severe consequences.

Insufficient signature information: A digital signature can be valid for multiple transactions, for example, if the sender sends money to multiple recipients through a proxy contract. If the signature does not provide the due information, a malicious recipient can replay the message multiple times to withdraw more money. This can be prevented by adding due information in each message, as outlined by Chen et al. [90].

6.3.3. Contract-Programming Vulnerabilities

Smart contracts are programmed either in a domain-specific programming language (DSL), such as Ethereum's Solidity, or in high-level languages, such as Go, Java, and Node.js. The latter has the advantage of reducing the developer's learning cost, as pointed out by Brotsis et al. [99]. Kushwaha [117] outlines vulnerabilities due to features of the EVM, such as immutability, missing orphan proof, vulnerabilities due to immutable bugs or mistakes, and Ether lost in the transfer. Chen et al. [90] highlight programming issues. Yang et al. [93] detail this by logical issues, centralization, volatile code, coding style, cost optimization, language-specific, control flow, math operations, inconsistency, data flow, and misconfiguration. Kushwaha et al. [117] additionally add uncontrolled resource consumption, external dependence, malleable entropy sources, insufficient authorization, improper validation, useless and repeated code, variable-sized parameters, improper access control, hash collision, unprotected Ether withdrawal, floating pragma, and delegate call. In the following, we briefly explain vulnerabilities found during our survey.

Delegatecall injection: This vulnerability was first observed during an attack against the Parity wallet. In order to facilitate code reuse, the EVM provides operation code (opcode, i.e., `delegatecall`) for inserting a callee contract's bytecode into the bytecode of the caller contract. As described by [90,116,120], malicious callee contracts can directly modify the state variables of the caller contracts. Similarly, fault injection might be possible, according to Hajdu et al. [128] and Muralidhara et al. [80]. Muralidhara et al. [80] additionally add code injection and structured query language (SQL) injection.

Erroneous visibility: If the function's visibility is not properly specified, it can provide unauthorized access [90,116,117,119,120,122]. This was first observed in an attack against the Parity wallet. As a countermeasure, Solidity makes it mandatory to specify function visibility starting with version 0.5.0 [135]. Similarly, Usman et al. [125] describe shadowing.

Authentication through tx.origin: `tx.origin` is a variable in Solidity and refers to the externally owned account (EOA) that initiated the transaction. It can be misused if it is applied to a contract and compromised by a phishing attack [90,103,117,120,126]. Hence, the authors of Solidity recommend not to use `tx.origin` for authorization [136].

Manipulated balance: This vulnerability can also be called forcing Ether to contracts, according to Chen et al. [90]. It occurs if a control-flow decision relies on the value of `this.balance` or `address(this).balance`, which can be used by an attacker to obtain the money. As a countermeasure, balance should not be used.

Unchecked call return values: This vulnerability is also called mishandled exceptions [90,114,126]. According to Chen et al. [90], it has two variants: gasless send and unchecked send. Similarly, Kushwaha et al. [117] outline improper exception handling and mishandling, and Ferreira Torres et al. [101] and Ashouri [102] unhandled exceptions.

Uninitialized storage pointer: This vulnerability was caused by Solidity's treatment of uninitialized compound local variables (overwriting from slot 0) and is eliminated by version 0.5.0 [135], according to Chen et al. [90].

Call to unknown: This vulnerability occurs if a function that does not exist in the target contract is called. Calling a non-existent function triggers the fallback function, posing a reentrancy attack risk, as described by Kushwaha et al. [117] and Chen et al. [90]. This was used in the Parity multi-signature wallet attacks [90,101,114,120,126]. Kushwaha et al. [117] generally describe fallback functions as a possible vulnerability. The erroneous constructor name vulnerability existed in Solidity until version 0.4.22. It allowed everyone to become the owner of the contract due to any incorrect name of a constructor function. It was first observed in the Rubixi contract, as outlined by Chen et al. [90].

Type casts: In Solidity, a contract can call another contract by directly referencing the callee's contract's instance. Here, a type cast issue can happen. A type cast is if one type is explicitly or implicitly converted to another. According to Solidity [136], types that do not occupy 32 bytes might contain "dirty higher order bits", which pose a risk if `msg.data` is accessed. Such a type cast can mislead EVM to run the attacker's contract [90,114,126].

Outdated compiler version: An outdated compiler version can contain bugs and thus make the compiled contract vulnerable, as outlined by Chen et al. [90] and Yang et al. [93].

Permission control: This vulnerability describes the improper use of `msg.origin` and `msg.sender`, as explained by He et al. [122]. `msg.origin` is the address of the EOA that originated the transaction; `msg.sender` is the address of the caller of the smart contract, which can be an EOA or a smart contract. With Ethereum Request for Comment (ERC)-4337 (account abstraction) [137], not only EOAs but also smart contracts have the ability to issue transactions on behalf of a user. This has the advantage of avoiding changes to the consensus-layer protocol.

6.3.4. Transaction Irregularities

Kushwaha et al. [117] outline the design features of the Ethereum blockchain that can cause vulnerabilities and irregularities, such as a malleable miner, a lack of transactional privacy, transaction ordering dependency, and an untrustworthy data feed.

Call-stack depth limit: The EVM's call-stack has a hard limit of 1024 frames. If a contract calls another contract, the call-stack depth increases by one. If the number of nested calls exceeds 1024, Solidity throws an exception and aborts the call. This can be misused by an attacker. The hard fork [138] for the Ethereum improvement proposal (EIP)-150 called Tangerine Whistle [139] re-defines the gas consumption rules of external calls to make it impossible to reach these 1024 frames. This vulnerability is described by Chen et al. [90].

Timestamp dependence: If a contract uses the `block.timestamp` as a part of a triggering condition when executing a critical operation or as the source of randomness, this could be manipulated by a malicious miner [69,90,103,116,117,119,120,122,125,126]. Generally, randomness is an issue, according to He et al. [122]. The initial private seeds, such as `block.number`, `block.timestamp`, `block.difficulty`, or `blockhash`, are fully controlled by miners. Consequently, a malicious miner can manipulate these values.

6.3.5. Countermeasures

We found various countermeasures. These comprise checking the smart contract and verifying the source code [131], logic analysis [101], security tools [103,116,117,120,122,124,126], flow analysis [115], visualization tools, disassembler and decompiler, linter, and miscellaneous tools [103], static and dynamic analysis [103,119,120], symbolic execution, formal verification [103], differential fuzzing [107], deep learning [109], and verification of identities [122], among others. Ivanov et al. [106] propose a taxonomy with static analysis, symbolic execution, fuzzing, formal analysis, machine learning methods, execution tracing, and transaction interception. Kissoon and Bekaroo [121] suggest open web application security project (OWASP) top 10, smart contract security verification standard (SCSVS), tools (code translation, static, and dynamic), fuzzing, and machine learning. Ahmadjee et al. [54] apply spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege (STRIDE) and MITRE. Van Landuyt et al. [48] try threat modeling, whereas Samanta et al. [49] apply game theory. Chen et al. [70] regard standardization, whereas Leng et al. [71] propose process security, data security, and infrastructure security.

6.4. Consensus Layer

As laid out by Brotsis et al. [99], the consensus protocol is the most critical component of a distributed ledger. According to the authors, Hyperledger Fabric and a crash fault tolerant (CFT) consensus protocol can be considered ideal in a confident network, such as an enterprise environment. However, non-confident networks also exisit, such as with cryptocurrencies. Following, we describe mining (see Section 6.4.1), timing (see Section 6.4.2), and other issues (see Section 6.4.3). Lastly, we briefly state countermeasures in Section 6.4.4.

6.4.1. Mining Issues

Several mining issues are explained. Specific attacks are the nothing-at-stake attack for PoS, pre-computation attack in PoW+PoS hybrid mode, the long-range attack in PoS, which is similar to the 51% attack (all described by Yu et al. [47]), the mining pool attack [49], and the private attack for PoS [60]. Furthermore, the flood and loot attack [94] and the wormhole attack [59] are outlined in the literature. The following vulnerabilities and attacks were named more often or are more generally applicable.

Honest mining assumption: Honest mining should be the most profitable mining strategy, which is not true. Selfish mining exploits the probabilistic finality vulnerability, the honest mining assumption vulnerability, and the rewards for uncle blocks vulnerability. Miners may withhold new blocks and selectively publish some blocks to earn an unfair share of rewards [51,52,67,76,80,81,90,103]. Wang et al. [76] additionally describe stubborn mining and the block-withholding attack. Miners can make a profit by taking bribes and changing their strategy to benefit the bribers [53,67,90]. Chen et al. [90] distinguish in-band bribery and out-of-band bribery. A similar attack is the fork after withholding (FAW) [67,76,91]. The attacker divides the computational power into two parts: one part mines the mining pool and the other part mines the victim pool. The attack combines selfish mining and block-withholding attacks. In the first case, the attacker finds a block in the innocent pool and uploads it normally. In the second case, the attacker finds a block in the target pool and discards it. Fore case three, the attacker finds a block, while others outside the victim pool also find one. The attacker submits their block to fork the blockchain. In the fourth case, others find a block and the attacker cannot do anything. In the first three cases, the attacker earns more money.

Misleading rewards: The rewards for the uncle block vulnerability refer to the uncle-rewarding mechanism, which was introduced to cope with the increase in stale blocks. This mechanism has the side effect of rewarding uncle blocks and, hence, selfish-mining. Ethereum adopted a PoW puzzle called Ethash [140] (former Ethereum's PoW mining algorithm), which is a modified version of the Dagger-Hashimoto algorithm. According to Chen et al. [90], miners are able to divide the task of searching for a puzzle solution into multiple smaller tasks, which are outsourced.

Probabilistic finality: This vulnerability in PoW and PoS protocols refers to the fact that the Ethereum blockchain can only achieve a probabilistic rather than a deterministic assurance that a new block will be finalized in the blockchain, according to Haugum et al. [90] and Chen et al. [59]. The deeper a block is contained in the blockchain, the more likely it is that it will not be reverted. The 51% attack [33,47,49,52,67,72–77,80,90,103,131], also called the majority attack, describes the fact when a single user or group of users gets control of more than 50% of the hashing power in a PoW blockchain. In a PoS blockchain, this could happen in specific cases, as outlined by Hao [75]. Successful attackers obtain the power to prevent new transactions from being completed, to reorder new transactions, to effectively rewrite sections of the blockchain, and to reverse transactions. The latter can lead to a problem known as double-spending [49,51,52,57,59,68,74,76,77,80,97,98,103]. The malicious user attempts to deceive the system by spending the same Bitcoin (BTC) more than once within a short timeframe. Other reasons for double-spending are race attacks, Finney attacks, and Vector76 attacks [51,67,97]. Iqbal and Matulevicius [68] add Sybil-based double-spending, PoS long-range attack, time advantage, Eclipse-based double-spending, border gateway protocol (BGP) hijacking, and a 0-confirmation race attack. As Ahmed et al. [33] state, larger networks are less prone to this attack and a trust-authority node can be introduced. Similarly, Fröhlich et al. [131] recommend using well-known networks.

Transaction vulnerability: In a flooding attack, the attacker issues numerous transactions, which leads to a flooding of the backlog of transactions since they wait to be confirmed, as outlined by Fröhlich et al. [131]. In a poisoning attack, as explained by Sato et al. [44] and Sharma and Shah [72], the attacker prepares a malicious or illegal file, embeds the file into the flexible space of a transaction, and broadcasts the transaction in the blockchain network. The malicious file is embedded into the blockchain through the mining process and then shared among the network participants. The file could contain privacy-related information, malware, or illegal content. Similarly, balance attacks, resource exhaustion attacks, and incorrect transaction attacks are described.

Verifier's dilemma: If the verification of a new transaction requires nontrivial computation efforts, miners are subject to attacks. If miners verify a computationally nontrivial transaction, they spend a significant amount of time. If miners accept the transaction without verification, then the blockchain may include an incorrect transaction. The resource exhaustion attack and the incorrect transaction attack, see Chen et al. [90], exploit this vulnerability, which can be mitigated by adding a gas limit and providing an incentive for honest miners. Wang et al. [88] describe a related issue for practical Byzantine fault tolerance (PBFT). A view is the period of time for which a given node is the primary. In a view change, the view is switched to another primary. The first attack requires a malicious primary to trigger the view change, whereas the second attack has the constraint of a timeout to trigger the view change.

6.4.2. Timing Issues

We found the following vulnerabilities related to timing and the transaction order.

Timing vulnerabilities: Side-chains are independent blockchains that employ their own consensus models to enhance transaction processing time. The transaction history is periodically updated on the main-chain and an old version is located on the side-chain. Hence, side-chains are vulnerable to timing attacks between the updates, as outlined by Haugum et al. [59]. Zhang et al. [66] describe a time-manipulation attack, which is applicable to proof-of-authority (PoA), another consensus algorithm.

Transaction order: Transaction ordering dependence (TOD) or front running refers to the state of blockchains depending on the execution order of transactions [90,102,103,110,116,119,120,126]. If the order of two transactions calling the same contract changes the final outcome, adversaries can exploit this property. Transactions are publicly broadcast on the network. Malicious actors can offer a higher gas price to have their transactions assembled into blocks sooner than the specific one. This vulnerability can be mitigated by hiding transactions or introducing a guard condition. According to Varun et al. [113], the following variants exist: displacement, insertion, and suppression. The authors apply machine learning for detection. Tjiam et al. [110] describe the sandwich attack as the simplest and most commonly encountered subtype of specialized frontrunning. The authors additionally state the variant of guaranteeing the transaction order to guarantee a profit from sandwich attacks. According to them, design paradigms against TOD are off-chain computation and tx batching, bypassing the memory pool (mempool), and commit-reveal on rollups. In a sandwich attack, as described by Wang et al. [92] and Tjam et al. [110], attackers take advantage of the transaction confirmation being delayed, causing financial losses for victims. The authors further state the countermeasures of no-profit and non-observable transactions. Similarly, in the liveliness attack based on Shah and Chopade [52] and Haugum et al. [59], the attackers delay the confirmation time of a target transaction.

6.4.3. Other Issues

According to Yu et al. [47], the security of blockchain depends on the strength of cryptographic encryption algorithms. The authors further explain that rainbow tables and quantum computing in the future might be issues. The latter is emphasized by Sharma and Shak [72]. In a replacement attack, the attacker tries to pass the data integrity check by replacing the challenged signature and block with an unchallenged block and signature, according to Ahmed et al. [33]. In a collusion attack, the attacker exploits false data injection [33,59,79]. According to Wang et al. [79], this can be against write-only, read-only, read-write, or delete-related transactions. Wijaya et al. [89] outline zero-mixin transactions and cascade effects, hard fork problems, and closed-set transaction attacks for Monero, whereas Sharma and Shak [72] and Islam et al. [73] generally name issues of forking. Tjiam et al. [110] explain oracle manipulation with the countermeasures of using time-weighted average prices and consulting m-of-n reporters within the oracle architecture.

6.4.4. Countermeasures

Several countermeasures can be found in the literature, ranging from a zero-block algorithm against selfish mining attacks without timestamps, randomly selecting mining groups, and increasing the minimal cost of attack to fuzzing, symbolic execution, and formal verification (see Sifra [123]) to detecting compromised blockchain nodes using a server-side authentication process and thwarting their activities before getting updated in the ledger (see Ajayi and Saadawi [78]).

6.5. Data Layer

Within the data layer, we found the following three vulnerabilities and issues.

Indistinguishable chains: Ethereum uses the elliptic curve digital signature algorithm (ECDSA) to sign transactions. Prior to the hard fork for EIP-155 [141] (simple replay attack protection), each transaction consisted of six fields. However, the signatures are not chain-specific. Hence, a transaction for one chain can be reused for another chain. With indistinguishable chains, a transaction could be used in another chain (cross-chain replay attack), as outlined by Chen et al. [90]. By including chainID, this vulnerability in Ethereum is eliminated.

Empty account in the state trie: An empty account is an account with zero nonce, zero balance, and no code or storage associated with it. It is functionally equivalent to a non-existing account, except that it is included in the Ethereum state trie. An attacker can create numerous empty accounts to cause a DoS attack. With the hard fork for EIP-161 [142], this vulnerability was eliminated, according to Chen et al. [90].

Trusted third parties: These trusted third parties belong to all layers, but the data layer is the most prominent. Fröhlich et al. [131] describe the issues of not updating wallets and not following laws. Swambo end Poinsot [83] outline access to the location of servers and the compromise of servers by software or human vulnerability. Yan et al. [56] generally point out that third parties can be adversaries. Hence, according to the authors, users should not rely on a single platform and make backups.

6.6. Network Layer

Generally, centralization and misconfiguration can be issues, according to Yang et al. [93]. Similarly, account hijacking is described at each layer, including the network layer. Brotsis et al. [99] outline the compromised membership service provider and identified endorsers at Hyperledger Fabric. In the following, we describe blockchain-specific (see Section 6.6.1) and general network issues (see Section 6.6.2). Lastly, we summarize countermeasures found in the literature.

6.6.1. Blockchain-Specific Issues

The following issues, vulnerabilities, and attacks are either specific to peer-to-peer (P2P) networks or blockchains.

Forgery attack: After acquiring the desired identity, the attacker can forge themselves as a legitimate entity, such as the verifier, according to Ahmed et al. [33]. Hu et al. [38] describe unforgeability as a countermeasure.

Unlimited node creation: According to Chen et al. [90], an attacker could create an unlimited number of nodes and use these nodes to monopolize the incoming and outgoing connections of victim nodes. Hence, isolating the victims from the other peers in the network. By restricting the node generation process, this vulnerability can be mitigated.

Uncapped incoming connections: Each node can have a total number of maximal peer connections, but there is no upper limit on incoming connections. Hence, this can be used by an attacker to establish incoming connections to a victim node so that they have no outgoing connections. By introducing an upper limit on incoming connections (see Geth v1.8), this vulnerability can be mitigated (see Chen et al. [90]).

Public/fixed peer selection: Both vulnerabilities have been discovered in the Geth client (using the Ethereum network with a modified Kademlia distributed hash table (DHT)) and have been fixed. If a node needs to locate a target node, it queries the 16 nodes in its bucket that are relatively close to the target node and asks each of these nodes to return the 16 IDs of their neighbors that are closer to the target node. The process iterates until the target node is reached. The mapping from node IDs to buckets is public. Hence, an attacker can craft node IDs that can land in a victim node's buckets and insert malicious node IDs into the victim node's routing table. The fixed peer selection refers to the client fetching the heads of randomly chosen buckets when selecting nodes from its routing table to establish outbound connections. As the nodes are sorted by activity, an attacker could make its node the first one, as outlined by Chen et al. [90].

Sole block synchronization: According to Chen et al. [90], this vulnerability allows an attacker to partition the Ethereum network without monopolizing the connections of a victim. Ethereum allows a client to synchronize with one other client at a time. If the other client is malicious and deliberately delays the synchronization, the blockchain at the victim is stalled and the victim rejects every subsequent block.

6.6.2. Network Issues

One network issue is message loss, as outlined by Ameen et al. [63]. Snegireva [67] and Muralidhara and Usha [80] name delay and DDoS, whereas AlFaw et al. [67] and Wang et al. [76] describe transaction malleability. Further network issues are routing attacks (see Zaghloul [97]), border gateway protocol (BGP) hijacking attacks (see Shah and Chopade [52] and Gupta et al. [103]), and DoS via route hijacking (see Tochner [65]). Following, we describe the most commonly stated attacks.

Impersonation attack: In an impersonation attack, the attacker tries to impersonate a trusted entity, as outlined by Ahmed et al. [33]. This could be a node in this case. Countermeasures are identification, authentication, and session keys.

Replay attack: In a replay attack, the attacker logs the session and resends the session or portions of it. Ahmed et al. [33] and Sharma and Shak [72] describe that the replay attack is one of the most frequent blockchain vulnerabilities. The authors explain that they are possible because the blockchain might be changed due to chain bifurcations or hard forks. In addition, Sharma and Shak [72] outline nonce tokens and timestamps as countermeasures. Based on EIP-155 [141], replay attacks have been an issue at Ethereum.

Sybil attack: The Sybil attack [33,50,52,58,59,67,68,76,97] refers to a malicious node with several valid IDs, which are used to block users' transactions and disconnect network connections. Iqbal and Matulevicius [68] describe the variations of breaking the consensus protocol attack, generating fake transactions attack, tampering nodes reputation, nodes isolation (partition) attack, routing table insertion attack, Sybil-based linking (deanonymization) attack, and Sybil-based DoS/DDoS attack. Countermeasures include a chain of trust mechanisms and consensus mechanisms, such as PoW and PoS, which are not prone to this attack. Zaghloul et al. [97] describe the limitation of outbound connections to single Internet protocol (IP) addresses.

Eclipse attack: The Eclipse attack is an attack against the process of establishing node information and connecting nodes in the blockchain. By acting as a MITM, the attacker intercepts communication and data exchange. Known countermeasures are checking IP addresses and using random IP addresses [51,59,72,73,76,80,97,103]. Ahmed et al. [33] outline the countermeasures of nonce and authentication against MITM attacks.

API exposure: Generally, application programming interfaces can be exposed or require no authentication. The remote procedure call (RPC) API exposure vulnerability was first observed during an attack against the Geth and Parity clients. The JSON-RPC of Ethereum clients provides APIs to communicate with the network. Although the API should only be available locally, the standard port 8545 for JSON-RPC was accessible remotely in the clients by default in the described attack. The vulnerability can be mitigated by configuring listening ports, adding access control to RPC calls, and generally practicing proper API design and configuration, as described by Chen et al. [90].

6.6.3. Countermeasures

Again, several generic countermeasures are stated. This includes attack-defense trees (see Eisentraut et al. [45]), machine learning, and analyzing commit messages (see Yi et al. [55]), as well as generally applying security features.

7. Results of the SSI Survey

Following, we briefly outline the findings of our literature survey concerning the security of SSI. Here, we initially found three publications. Since Rayhan Ahmed et al. [33] mainly describe the current state of blockchain and SSI and outline ten attacks on the consensus and network layers, we described in Sections 6.4 and 6.6, we disregard the publication in this section. Hence, we focus on Naik et al. [32] and Grüner et al. [34]. Naik et al. [32] first identify assets and potential attacks before generating an attack tree for each identified attack. Grüner et al. [34] apply STRIDE to the various components of SSI. In the following, we summarize and combine the results of both approaches.

7.1. Human and Credential Exchange Layer

Naik et al. [32] have identified three attacks and generated for each of them an attack tree. In order to obtain fake credentials, a user may either create fake credentials at an issuer, spoof the issuer, amend issued credentials, or steal credentials. To create fake credentials at an issuer, the attacker either has to obtain admin credentials (malware or social engineering) or store fake credential data (accessing the issuer host or updating data). For spoofing, three variants exist: creating a replica issuer host, publishing a fake issuer DID to the SSI network, or reducing trust in the original issuer (by applying either the Sybil or Eclipse attack). To amend issued credentials, the attacker has to obtain the issuer's private key (i. e., access the issuer host and locate the private key, which could be done by malware or a social engineering attack) or sign the updated credential in the wallet. To steal credentials, the attacker has to either steal a wallet (steal the phone or attack the user's cloud storage) or impersonate the user (obtain credentials and request credentials).

Naik et al. [32] additionally describe the threat of obtaining personal data. Here, the authors identify three vectors: unauthorized access to the user's wallet, credential creep, and background data attack. To get unauthorized access to the wallet, the attacker needs to receive the user credentials (using malware or a social engineering attack) and access the wallet (stealing the phone or gaining remote access to the wallet). Credential creep is possible by requesting additional data or user profiling (multiple verifier requests for credentials and linking DID to identify the user). In a background data attack, the attacker has to obtain a sensitive dataset and link data via a verifier request.

7.2. Communication Layer

Grüner et al. [34] apply the STRIDE model to the user agent. Thereby, the authors recognize spoofing (acquiring the private key, stealing or covertly accessing the user agent device, and exploiting the recovery mechanism of the identity), tampering with the user agent's data, repudiating identity actions (deliberately disclosing the private key, revoking an identity, and deliberately losing the user agent device), revealing confidential identity information, denying identity actions (stealing or breaking the user agent device, deleting the private key, and exploiting identity revocation), and elevation of privileges on the user agent (not applicable). In addition, the VC is stored, which again can be manipulated. Here, the authors determine spoofing (self-attested claims), tampering (changing VC value), repudiating (deleting the VC), revealing confidential VC information (gaining unauthorized access and requesting unnecessary data), denying VC store serviceability, and elevating privileges on the VC store.

Grüner et al. [34] regard the organizational agent. By applying STRIDE, they identify spoofing (misusing an identity), tampering (manipulating a configuration), repudiation (illegitimate revocing a VC), revealing confidential identity information (not applicable), denying identity actions, and elevating privileges (taking-over a role). Similarly, the authors analyze the trust and data stores. For spoofing and tampering, the authors recognize circumventing VC verification and manipulating trusted issuers; for revealing confidential information, they state the disclosure of trusted issuers. The other categories are not SSI-specific, according to the authors.

By utilizing STRIDE, Grüner et al. [34] identify in the area of communication the issues of spoofing and tampering (spoofing communication partner), repudiation (disputing a message), and revealing confidential information (intercepting traffic).

7.3. Repository Technology Layer

Grüner et al. [34] analyze the identity holder, verifier, and issuer nodes. Here, the authors find spoofing (propagating a forged message), tampering with the node (manipulating state and configuration), and denying node serviceability (resetting or closing connections and flooding connections). The other categories either are not applicable or have no specific threats to SSI.

Naik et al. [32] analyze DoS attacks. The authors identify three variants: deny services to the host, BDoS, and disrupt the SSI operational framework. In order to deny services to the host, the various entities could be flooded. The authors further name volume-based attacks, protocol-based attacks, application layer attacks, buffer overflow attacks, and radio frequency (RF) interference attacks. To target the blockchain, the following attacks are stated: flood blockchain nodes, resource depletion attacks, increase fake validators, discourage validators participation, Sybil attacks, and Eclipse attacks. Lastly, to disrupt the SSI operational framework, the governance framework or the regulatory framework could be interrupted.

Additionally, Grüner et al. [34] evaluate verifiable data registries (VDRs). Here, the authors recognize spoofing, tampering, and repudiation by exploiting smart contract vulnerabilities. The serviceability could be denied by deactivating VDR smart contracts and manipulating the blockchain configuration. Privileges could be elevated by taking over the VDR owner, holder, or issuer role. The disclosure of confidential information is not applicable, according to the authors.

8. Application and Discussion of the Results

So far, we have summarized the findings of our literature survey on blockchain applications in genral (see Section 6) and SSI (see Section 7). Thereby, we have answered the research questions RQ1 and RQ2. However, since blockchain applications on cryptocurrencies (the most common application in the literature found) are different from those for SSI, we first apply these results to SSI in Section 8.1. For this step, we still use the layers of blockchain. The application of the results to SSI should answer RQ3. Next, in Section 8.2, we summarize the findings and combine them with the survey on SSI threats. Lastly, we outline future work based on the layers of SSI in Section 8.3 and, thereby, answer RQ4.

8.1. Application on SSI

We included 101 publications in our literature survey concerning threats to elements of SSI, such as wallets, DLT, humans, and smart contracts. We categorized the threats based on the blockchain layers. During the analysis of the literature survey, we noticed that most publications concern cryptocurrencies, especially Ethereum. Ethereum uses smart contracts to transfer ETH. Originally, Ethereum applied PoW and changed to PoS in 2022, which reduces energy consumption [143]. Nonetheless, cryptocurrencies are different from SSI, which may have consequences for the applied technologies and, hence, threats. In the following, we discuss the application of the threats to SSI layer by layer.

8.1.1. Human Layer

Similarly to cryptocurrencies, the SSI stack includes a human layer, in which the end-user uses their wallet to access services. Additionally, humans administrate servers, interact within the service desk, and provide the governance layer, among others. The categories of social engineering, human errors, and wallet threats also apply to SSI. However, successful attacks have other consequences for the users since SSI handles identity-related data. Furthermore, the described issues are more general, especially concerning social engineering.

8.1.2. Application Layer

Although smart contracts can be applied in SSI, they are, at least currently, most likely not used in SSI infrastructures. The literature focuses on Solidity, the programming language of Ethereum. Based on the statistics provided in Section 5, we notice a decline in publications on that topic. Hence, most threats outlined in Section 6.3 could be applied to SSI but are less relevant. One exception is the threat of transaction irregularities, which could work on SSI. Following, the threats on the application layer have to be regarded.

8.1.3. Consensus Layer

Most publications within the consensus layer focus on PoW and PoS, either explicitly or implicitly. However, PoW and PoS are typically not chosen for SSI. For example, Hyperledger Indy uses a variant of redundant Byzantine fault tolerance (RBFT) called Plenum [144] as the consensus algorithm, whereas Hyperledger Fabric lets the administrators choose [145]. Following, the literature found for our survey and, hence, the threats, might not be applicable to SSI.

8.1.4. Data Layer

The rare findings concerning the data layer can probably be applied to SSI. However, there might be more threats, which still have to be analyzed.

8.1.5. Network Layer

The vulnerabilities and attacks found in the network layer can be divided into blockchain- or P2P-specific issues (for P2P, see, for example, Schäfer et al. [146]) and general network issues. The blockchain-specific and network-related items can be applied to SSI. However, some issues, such as impersonation, Sybil attack, and Eclipse attack, might be mitigated due to governance and configuration.

8.2. Summary

We identified several threats (RQ1) and some countermeasures (RQ2). Based on the discussion in the previous section, we notice that not every threat and attack can be applied to SSI, whereas some descriptions are of a generic nature (RQ3). Although the human layer can be applied to SSI, it has to be regarded in more detail. The threats of the application layer are most likely not applicable to SSI. Nonetheless, SSI has applications, such as wallets and agents, and VC that are sent between entities. The first results can be found in the literature for SSI, but more work is required. The consensus algorithms of cryptocurrencies outlined in the literature and SSI diverge. Hence, those applied for SSI, such as RBFT, and their implementations have to be evaluated. Although the threats of the data layer can be applied, we found only a few threats in the literature. Thus, this might require future work. Lastly, the threats and attacks concerning the network layer can be applied to SSI. Some might be mitigated by governance, which is added as a layer in the following section. Here, we also found the most known threats to SSI. One issue with the analysis of threats to SSI is that the area is still progressing. Hence, the threats are rather generic and not specific for one or a few implementations. If SSI should be enrolled in a certain use case, the threats have to be evaluated for the setting, including the chosen implementations. For example, threats to Hyperledger Indy might be different from those to Hyperledger Fabric, and the type of wallet (smartphone app, cloud, or hybrid wallet) also plays a role. Since several threats cannot be applied to SSI directly, the countermeasures have to be regarded specifically for SSI.

8.3. Future Work

Based on our findings, we identify the following topics for future work per layer (RQ4). In general, the analysis of threats should include current implementations.

8.3.1. Governance Layer

Most publications focus on technology, but not on governance. However, governance provides a framework for the operation of blockchain and, hence, SSI. This includes a general framework for the operation, a trust framework describing the interactions and trust of the participants, an agent and wallet framework, suggesting specific types and versions, and a utility framework, focusing on methods and similar. Governance can have an impact on security by, for example, recommending vulnerable versions of agents. Consequently, more work is required on the topic of governance, including security management. Within security management, security incidents have to be taken care of, among others.

8.3.2. Human Layer

We found five publications out of 101 focusing on human aspects of security related to blockchain. Ten further publications included blockchain wallets and their security. Humans and wallets are considered in Section 7, analyzing the threats to SSI. Nonetheless, several aspects still have to be explored. For example, studies may indicate how to mitigate phishing in this context and what role the user interface (UI) plays. In Section 3, we evaluated several SSI wallets on the market. Although they seem to follow some design rules, we recognized some issues with the UI and security. Further studies may provide a design guide and awareness-raising designs to mitigate threats. Another question is how to raise the security and privacy awareness of users and trigger secure actions. The importance of this question is emphasized by Graux [147]. Also, the implications of SSI for social engineering attacks in general can be analyzed, not only focusing on phishing.

8.3.3. Credential Exchange Layer

Based on our results in Section 3, the threats to wallets should be analyzed more thoroughly. This includes static and dynamic analysis of SSI wallets for different OSs and not only Android. Here, a test suite could partially automate the analysis. The results may lead to best practice guides and more secure wallets. These can be followed by developers and consortia. Another complex topic is the supply chain in this respect. Wallets, like several other software products, use various libraries, which could be malicious or at least buggy. A methodology to analyze these dependencies could help.

8.3.4. Communication Layer

The implementations for data exchange and their protocols should be analyzed. The early formal analysis for OpenID4VP over Bluetooth Low Energy (BLE) by Felix Linker (Provided within the standardization mailing list.) led to improvements during the standardization process. This approach could be adapted to various protocols. In addition, implementations can be analyzed, for example, with static and dynamic code analysis. Although various publications focus on consensus algorithms, those applied in the SSI context are typically not regarded. Hence, this is another topic for future work. We additionally searched for the security of zero-knowledge proofs, which can be applied in SSI, and especially their implementations. However, we found no publications on that topic.

8.3.5. Repository Technology Layer

Although several vulnerabilities and issues in the DID networks layer are already known, a more thorough analysis is required. This is especially true since differences in blockchain for cryptocurrencies exist, governance plays a role in this layer, and the threats may depend on the actual implementation. The analysis of the vulnerabilities could involve the verification within a testbed.

9. Conclusions

Identity management is a critical component in developing and deploying digital services in various fields. With the increasing digitalization, the demand for reliable and secure identity management is greater than ever. Currently, a handful of commercial providers dominate the field and, thereby, are in a position to aggregate metadata and receive a profile of user activities. To give control back to the users, SSI was established. With the progress of the scientific approach and the partial adoption planned in the new eIDAS regulation, security becomes more relevant. In order to provide first insights, we analyzed current SSI wallets on the market as motivation. Then, we conducted a literature survey on the threats, vulnerabilities, and security of SSI and its elements. We applied the gathered results to SSI if possible and discussed future work. Based on our survey, several aspects related to the threats to SSI are still uncovered and require more research. This is essential before adopting SSI for identity management.

Author Contributions: Conceptualization, D.P., M.G. and W.H.; methodology, D.P.; validation, D.P., M.G. and W.H.; investigation, D.P.; data curation, D.P.; writing—original draft preparation, D.P.; writing—review and editing, D.P., M.G. and W.H.; visualization, D.P. and M.G.; supervision, W.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

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

Abbreviations

The following abbreviations are used in this manuscript:

2FA	Two-factor authentication
ADB	Android Debug Bridge
API	Application programming interface
APK	Android package
BDoS	Blockchain denial-of-service
BEC	BeautyChain
BFT	Byzantine fault tolerance
BGP	Border gateway protocol
BLE	Bluetooth Low Energy
BTX	Bitcoin
CFT	Crash fault tolerant
CWE	Common weakness enumeration
DAO	Decentralized autonomous organization
DDoS	Distributed denial-of-service
DES	Data encryption standard
DEX	Dalvic executable format
DHT	Distributed hash table
DID	Decentralized identifier
DLT	Distributed ledger technology
DoS	Denial-of-service
DSL	Domain-specific programming language
ECDSA	Elliptic curve digital signature algorithm

eIDAS	electronic Identification, Authentication and Trust Services
EIP	Ethereum improvement proposal
EOA	Externally owned account
ERC	Ethereum Request for Comment
ETH	Ether
EU	European Union
EVM	Ethereum virtual machine
FAW	Fork after withholding
HTLC	hash time locked contract
IP	Internet protocol
JAR	Java archive
JSON	Java Script Object Notation
MD5	Message-digest algorithm 5
mempool	Memory pool
MFA	Multi-factor authentication
MITM	Human-in-the-middle
OIDC	OpenID Connect
opcode	operation code
OpenID4VP	OpenID for Verifiable Presentations
OS	Operating system
OSI	Open Systems Interconnection
OWASP	Open web application security project
P2P	Peer-to-peer
PBFT	Practical Byzantine fault tolerance
PCB	Printed circuit board
PIN	Personal identification number
PKI	Public key infrastructure
PoA	Proof-of-authority
PoS	Proof-of-stake
PoW	Proof-of-work
QR	Quick response
RBFT	Redundant Byzantine fault tolerance
RF	Radio frequency
RPC	Remote procedure call
SCSVS	Smart contract security verification standard
SHA	Secure hash algorithm
SIM	Subscriber identity module
SIOP	Self Issued OpenID Provider
SMS	Short message service
SQL	Structured query language
SSI	Self-sovereign identity
SSL	Secure sockets layer
STRIDE	Spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege
TCP	Transmission control protocol
TLS	Transport layer security
TOD	Transaction-ordering dependency
tx	Transaction
UI	User interface
VDR	Verifiable data registries
VC	Verifiable credential
VPN	Virtual private network
XML	extensible markup language

References

1. European Parliament and Council. Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on Electronic Identification and Trust Services for Electronic Transactions in the Internal Market and Repealing Directive 1999/93/EC. Available online: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32014R0910 (accessed on 6 November 2023).
2. Lips, S.; Vinogradova, N.; Krimmer, R.; Draheim, D. Re-Shaping the EU Digital Identity Framework. In Proceedings of the 23rd Annual International Conference on Digital Government Research (DG.O), Virtual, 15–17 June 2022; ACM Digital Library: New York, NY, USA, 2022; pp. 13–21. [CrossRef]
3. Lukkien, B.; Bharosa, N.; De Reuver, M. Barriers for Developing and Launching Digital Identity Wallets. In Proceedings of the 24th Annual International Conference on Digital Government Research (DG.O), Gdansk, Poland, 11–14 July 2023; pp. 289–299. [CrossRef]
4. European Commission. European Digital Identity Architecture and Reference Framework—Outline. 2023. Available online: https://digital-strategy.ec.europa.eu/en/library/european-digital-identity-architecture-and-reference-framework-outline (accessed on 19 October 2023).
5. Sakimura, N.; Bradley, J.; Jones, M.B.; de Medeiros, B.; Mortimore, C. *OpenID Connect Core 1.0 Incorporating Errata Set 1*; Standard; OpenID Foundation: San Ramon, CA, USA, 2014. Available online: https://openid.net/specs/openid-connect-core-1_0.html (accessed on 19 October 2023).
6. Yasuda, K.; Jones, M.B.; Lodderstedt, T. *Self-Issued OpenID Provider v2*; Standard; OpenID Foundation: San Ramon, CA, USA, 2023. Available online: https://openid.net/specs/openid-connect-self-issued-v2-1_0.html (accessed on 19 October 2023).
7. Terbu, O.; Lodderstedt, T.; Yasuda, K.; Looker, T. *OpenID for Verifiable Presentations—Draft 18*; OpenID Foundation: San Ramon, CA, USA, 2023. Available online: https://openid.net/specs/openid-4-verifiable-presentations-1_0.html (accessed on 19 October 2023).
8. Hedberg, R.; Jones, M.B.; Solberg, A.A.; Bradley, J.; De Marco, G.; Dzhuvinov, V. *OpenID Federation 1.0—Draft 31*; OpenID Foundation: San Ramon, CA, USA, 2023.
9. Schwalm, S. The possible impacts of the eIDAS 2.0 digital identity approach in Germany and Europe. In Proceedings of the 10th Open Identity Summit (OID), Heilbronn, Germany, 15–16 June 2023; Gesellschaft für Informatik e.V.: Bonn, Germany, 2023; pp. 109–120. [CrossRef]
10. Kaushal, P.K.; Bagga, A.; Sobti, R. Evolution of bitcoin and security risk in bitcoin wallets. In Proceedings of the 1st International Conference on Computer, Communications and Electronics (Comptelix), Jaipur, India, 1–2 July 2017; pp. 172–177. [CrossRef]
11. Allen, C. The Path to Self-Sovereign Identity. 2016. Available online: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html, (accessed on 19 October 2023).
12. Preukschat, A.; Reed, D. *Self-Sovereign Identity*; Manning Publications: Shelter Island, NY, USA, 2021.
13. Mühle, A.; Grüner, A.; Gayvoronskaya, T.; Meinel, C. A Survey on Essential Components of a Self-Sovereign Identity. *Comput. Sci. Rev.* **2018**, *30*, 80–86. [CrossRef]
14. Sovrin Glossary V3. 2019. Available online: https://docs.google.com/document/d/1gfIz5TT0cNp2kxGMLFXr19x1uoZsruUe_0glHst2fZ8 (accessed on 19 October 2023).
15. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Whitepaper, Bitcoin. 2008. Available online: http://www.cryptovest.co.uk/resources/Bitcoin%20paper%20Original.pdf (accessed on 19 October 2023).
16. Jain, A.; Arora, S.; Shukla, Y.; Patil, T.; Sawant-Patil, S. Proof of Stake with Casper the Friendly Finality Gadget Protocol for Fair Validation Consensus in Ethereum. *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.* **2018**, *3*, 291–298.
17. Wöhrer, M.; Zdun, U. Design Patterns for Smart Contracts in the Ethereum Ecosystem. In Proceedings of the 9th IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018; pp. 1513–1520. [CrossRef]
18. European Blockchain Associaton. SSI Wallets. 2021. Available online: https://europeanblockchainassociation.org/ssi-wallets/ (accessed on 19 October 2023).
19. Main Incubator GmbH. Lissi—Identity Wallet an Identity Management Solutions. 2023. Available online: https://lissi.id/ (accessed on 19 October 2023).
20. Verimi. Verimi ID Wallet—Your Digital Wallet. 2023. Available online: https://verimi.de/en/ (accessed on 19 October 2023).
21. iGrant.io. iGrant.io—Your Data, Your Choice. 2023. Available online: https://igrant.io (accessed on 19 October 2023).
22. Esatus AG. Esatus AG—Enforcing Information Security. 2023. Available online: https://esatus.com/ (accessed on 19 October 2023).
23. Validated ID. VIDwallet—Regain Control of Your Digital Identity. 2023. Available online: https://www.validatedid.com/en/vidchain/vidwallet (accessed on 19 October 2023).
24. Jolocom. We Create Solutions for the Future of Digital Identity. 2023. Available online: https://jolocom.io (accessed on 19 October 2023).
25. Gataca. Trusted Digital Identities Made Easy. 2023. Available online: https://www.gataca.io (accessed on 19 October 2023).
26. Google. Analyze Your Build with the APK Analyzer. 2023. Available online: https://developer.android.com/studio/debug/apk-analyzer (accessed on 19 October 2023).

27. RedHunt Labs. APKHunt | OWASP MASVS Static Analyzer. 2023. Available online: https://github.com/Cyber-Buddy/APKHunt (accessed on 19 October 2023).
28. Uddin, M.S.; Mannan, M.; Youssef, A. Horus: A Security Assessment Framework for Android Crypto Wallets. In *Security and Privacy in Communication Networks: 17th EAI International Conference, SecureComm 2021, Virtual Event, 6–9 September 2021*; Garcia-Alfaro, J., Li, S., Poovendran, R., Debar, H., Yung, M., Eds.; Springer: Cham, Switzerland, 2021; pp. 120–139.
29. Google. Android Studio. 2023. Available online: https://developer.android.com/studio (accessed on 19 October 2023).
30. Teuschel, M.; Pöhn, D.; Grabatin, M.; Dietz, F.; Hommel, W.; Alt, F. 'Don't Annoy Me With Privacy Decisions!'—Designing Privacy-Preserving User Interfaces for SSI Wallets on Smartphones. *IEEE Access* **2023**, *11*, 131814–131835. [CrossRef]
31. Page, M.J.; McKenzie, J.E.; Bossuyt, P.M.; Boutron, I.; Hoffmann, T.C.; Mulrow, C.D.; Shamseer, L.; Tetzlaff, J.M.; Akl, E.A.; Brennan, S.E.; et al. The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ* **2021**, *88*, 105906 [CrossRef]
32. Naik, N.; Grace, P.; Jenkins, P. An Attack Tree Based Risk Analysis Method for Investigating Attacks and Facilitating Their Mitigations in Self-Sovereign Identity. In Proceedings of the 7th IEEE Symposium Series on Computational Intelligence (SSCI), Orlando, FL, USA, 5–7 December 2021; pp. 1–8. [CrossRef]
33. Ahmed, M.R.; Islam, A.K.M.M.; Shatabda, S.; Islam, S. Blockchain-Based Identity Management System and Self-Sovereign Identity Ecosystem: A Comprehensive Survey. *IEEE Access* **2022**, *10*, 113436–113481. [CrossRef]
34. Grüner, A.; Mühle, A.; Lockenvitz, N.; Meinel, C. Analyzing and comparing the security of self-sovereign identity management systems through threat modeling. *Int. J. Inf. Secur.* **2023**, *22*, 1231–1248. [CrossRef]
35. Wilusz, D.; Wójtowicz, A. Security Analysis of Transaction Authorization Methods for Next Generation Electronic Payment Services. In Proceedings of the 3rd International HCI Conference for Cybersecurity, Privacy and Trust (HCI-CPT), Virtual, 24–29 July 2021; Moallem, A., Ed.; Springer: Cham, Switzerland, 2021; pp. 103–119.
36. Do, T.L.; Tran, M.K.; Nguyen, H.H.; Tran, M.T. Potential Threat of Face Swapping to EKYC with Face Registration and Augmented Solution with Deepfake Detection. In Proceedings of the 8th International Conference on Future Data and Security Engineering (FDSE), Virtual, 24–26 November 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 293–307. [CrossRef]
37. Ohm, M.; Plate, H.; Sykosch, A.; Meier, M. Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks. In Proceedings of the 17th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), Lisbon, Portugal, 24–26 June 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 23–43. [CrossRef]
38. Hu, Y.; Wang, S.; Tu, G.H.; Xiao, L.; Xie, T.; Lei, X.; Li, C.Y. Security Threats from Bitcoin Wallet Smartphone Applications: Vulnerabilities, Attacks, and Countermeasures. In Proceedings of the 11th ACM Conference on Data and Application Security and Privacy (CODASPY), Virtual, 26–28 April 2021; ACM Digital Library: New York, NY, USA, 2021; pp. 89–100. [CrossRef]
39. Das, P.; Erwig, A.; Faust, S.; Loss, J.; Riahi, S. The Exact Security of BIP32 Wallets. In Proceedings of the 28th ACM SIGSAC Conference on Computer and Communications Security (CCS), Virtual, 15–19 November 2021; ACM Digital Library: New York, NY, USA, 2021; pp. 1020–1042. [CrossRef]
40. Dabrowski, A.; Pfeffer, K.; Reichel, M.; Mai, A.; Weippl, E.R.; Franz, M. Better Keep Cash in Your Boots—Hardware Wallets Are the New Single Point of Failure. In Proceedings of the 1st ACM CCS Workshop on Decentralized Finance and Security (DeFi), Virtual, 15 November 2021; ACM Digital Library: New York, NY, USA, 2021; pp. 1–8. [CrossRef]
41. He, D.; Li, S.; Li, C.; Zhu, S.; Chan, S.; Min, W.; Guizani, N. Security Analysis of Cryptocurrency Wallets in Android-Based Applications. *IEEE Netw.* **2020**, *34*, 114–119. [CrossRef]
42. Li, C.; He, D.; Li, S.; Zhu, S.; Chan, S.; Cheng, Y. Android-based Cryptocurrency Wallets: Attacks and Countermeasures. In Proceedings of the 2nd IEEE International Conference on Blockchain (Blockchain), Rhodes, Greece, 2–6 November 2020; pp. 9–16. [CrossRef]
43. Park, D.; Choi, M.; Kim, G.; Bae, D.; Kim, H.; Hong, S. Stealing Keys from Hardware Wallets: A Single Trace Side-Channel Attack on Elliptic Curve Scalar Multiplication without Profiling. *IEEE Access* **2023**, *11*, 44578–44589. [CrossRef]
44. Sato, T.; Imamura, M.; Omote, K. Threat Analysis of Poisoning Attack Against Ethereum Blockchain. In Proceedings of the 13th IFIP WG 11.2 International Conference on Information Security Theory and Practice (WISTP), Paris, France, 11–12 December 2019; Laurent, M., Giannetsos, T., Eds.; Springer: Cham, Switzerland, 2020; pp. 139–154.
45. Eisentraut, J.; Holzer, S.; Klioba, K.; Křetínský, J.; Pin, L.; Wagner, A. Assessing Security of Cryptocurrencies with Attack-Defense Trees: Proof of Concept and Future Directions. In Proceedings of the 18th International Colloquium on Theoretical Aspects of Computing (ICTAC), Nur-Sultan, Kazakhstan, 8–10 September 2021; Cerone, A., Ölveczky, P.C., Eds.; Springer: Cham, Switzerland, 2021; pp. 214–234.
46. Raikwar, M.; Gligoroski, D. DoS Attacks on Blockchain Ecosystem. In Proceedings of the International Parallel Processing Workshops (Euro-Par), Lisbon, Portugal, 30–31 August 2021; Revised Selected Papers; Chaves, R., Heras, D.B., Ilic, A., Unat, D., Badia, R.M., Bracciali, A., Diehl, P., Dubey, A., Sangyoon, O., Scott, S.L., et al., Eds.; Springer: Cham, Switzerland, 2022; pp. 230–242.
47. Yu, G.; Ni, C.; Liu, T. Research on Blockchain Security Risk Analysis and Coping Strategies. In Proceedings of the 3rd International Conference on Big Data and Security (ICBDS), Shenzen, China, 26–28 November 2021; Tian, Y., Ma, T., Khan, M.K., Sheng, V.S., Pan, Z., Eds.; Springer: Singapore, 2022; pp. 230–242.

48. Van Landuyt, D.; Sion, L.; Vandeloo, E.; Joosen, W. On the Applicability of Security and Privacy Threat Modeling for Blockchain Applications. In Proceedings of the Computer Security—ESORICS 2019 International Workshops, CyberICPS, SECPRE, SPOSE, and ADIoT, Luxembourg, 26–27 September 2019; Revised Selected Papers; Katsikas, S., Cuppens, F., Cuppens, N., Lambrinoudakis, C., Kalloniatis, C., Mylopoulos, J., Antón, A., Gritzalis, S., Pallas, F., Pohle, J., et al., Eds.; Springer: Cham, Switzerland, 2020; pp. 195–203.
49. Samanta, A.K.; Sarkar, B.B.; Chaki, N. Quantified Analysis of Security Issues and Its Mitigation in Blockchain Using Game Theory. In Proceedings of the 3rd International Conference on Computational Intelligence in Communications and Business Analytics (CICBA), Santiniketan, India, 7–8 January 2021; Revised Selected Papers; Dutta, P., Mandal, J.K., Mukhopadhyay, S., Eds.; Springer: Cham, Switzerland, 2021; pp. 3–19.
50. Kedziora, M.; Kozlowski, P.; Jozwiak, P. Security of Blockchain Distributed Ledger Consensus Mechanism in Context of the Sybil Attack. In Proceedings of the 33rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems: Trends in Artificial Intelligence Theory and Applications, Artificial Intelligence Practices (IEA/AIE), Kitakyushu, Japan, 22–25 September 2020; Fujita, H., Fournier-Viger, P., Ali, M., Sasaki, J., Eds.; Springer: Cham, Switzerland, 2020; pp. 407–418.
51. Dholey, M.K.; Ganguly, A. Major Challenges and Threats of Blockchain Technology. In Proceedings of the 1st International Symposium on Artificial Intelligence (ISAI), Haldia, India, 17–22 February 2022; Revised Selected Papers; Sk, A.A., Turki, T., Ghosh, T.K., Joardar, S., Barman, S., Eds.; Springer: Cham, Switzerland, 2022; pp. 96–108.
52. Shah, P.; Chopade, M. Blockchain Security: A Systematic Review. In Proceedings of the 4th International Conference on Futuristic Trends in Networks and Computing Technologies (FTNCT), Ahmedabad, India, 10–11 December 2022; Singh, P.K., Wierzchoń, S.T., Chhabra, J.K., Tanwar, S., Eds.; Springer: Singapore, 2022; pp. 969–980.
53. Sun, H.; Ruan, N.; Su, C. How to Model the Bribery Attack: A Practical Quantification Method in Blockchain. In Proceedings of the 25th European Symposium on Research in Computer Security (ESORICS), Guildford, UK, 14–18 September 2020; Chen, L., Li, N., Liang, K., Schneider, S., Eds.; Springer: Cham, Switzerland, 2020; pp. 569–589.
54. Ahmadjee, S.; Mera-Gómez, C.; Bahsoon, R.; Kazman, R. A Study on Blockchain Architecture Design Decisions and Their Security Attacks and Threats. *ACM Trans. Softw. Eng. Methodol.* **2022**, *31*, 36e. [CrossRef]
55. Yi, X.; Wu, D.; Jiang, L.; Fang, Y.; Zhang, K.; Zhang, W. An Empirical Study of Blockchain System Vulnerabilities: Modules, Types, and Patterns. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), Singapore, 14–18 November 2022; ACM Digital Library: New York, NY, USA, 2022; pp. 709–721. [CrossRef]
56. Yan, K.; Zhang, J.; Liu, X.; Diao, W.; Guo, S. Bad Apples: Understanding the Centralized Security Risks in Decentralized Ecosystems. In Proceedings of the 32nd ACM Web Conference (WWW), Austin, TX, USA, 30 April–4 May 2023; ACM Digital Library: New York, NY, USA, 2023; pp. 2274–2283. [CrossRef]
57. Saad, M.; Chen, S.; Mohaisen, D. SyncAttack: Double-Spending in Bitcoin without Mining Power. In Proceedings of the 27th ACM SIGSAC Conference on Computer and Communications Security (CCS), Virtual, 15–19 November 2021; ACM Digital Library: New York, NY, USA, 2021; pp. 1668–1685. [CrossRef]
58. Serena, L.; D'Angelo, G.; Ferretti, S. Implications of Dissemination Strategies on the Security of Distributed Ledgers. In Proceedings of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems (CryBlock), London, UK, 25 September 2020; ACM Digital Library: New York, NY, USA, 2020; pp. 65–70. [CrossRef]
59. Haugum, T.; Hoff, B.; Alsadi, M.; Li, J. Security and Privacy Challenges in Blockchain Interoperability—A Multivocal Literature Review. In Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering (EASE), Gothenburg, Sweden, 13–15 June 2022; ACM Digital Library: New York, NY, USA, 2022; pp. 347–356. [CrossRef]
60. Azouvi, S.; Cappelletti, D. Private Attacks in Longest Chain Proof-of-Stake Protocols with Single Secret Leader Elections. In Proceedings of the 3rd ACM Conference on Advances in Financial Technologies (AFT), Arlington, VA, USA, 26–28 September 2021; ACM Digital Library: New York, NY, USA, 2021; pp. 170–182. [CrossRef]
61. Li, X.; Luo, H.; Duan, J. Security Analysis of Sharding in Blockchain with PBFT Consensus. In Proceedings of the 4th International Conference on Blockchain Technology (ICBCT), Shanghai, China, 25–27 March 2022; ACM Digital Library: New York, NY, USA, 2022; pp. 9–14. [CrossRef]
62. Mirkin, M.; Ji, Y.; Pang, J.; Klages-Mundt, A.; Eyal, I.; Juels, A. BDoS: Blockchain Denial-of-Service. In Proceedings of the 26th ACM SIGSAC Conference on Computer and Communications Security (CCS), Virtual, 9–13 November 2020; ACM Digital Library: New York, NY, USA, 2020; pp. 601–619. [CrossRef]
63. Ameen, T.; Sankagiri, S.; Hajek, B. Blockchain Security When Messages Are Lost. In Proceedings of the 1st ACM Workshop on Developments in Consensus (ConsensusDay), Los Angeles, CA, USA, 7 November 2022; ACM Digital Library: New York, NY, USA, 2022; pp. 1–14. [CrossRef]
64. Huang, H.; Kong, W.; Zhou, S.; Zheng, Z.; Guo, S. A Survey of State-of-the-Art on Blockchains: Theories, Modelings, and Tools. *ACM Comput. Surv.* **2021**, *54*, 1–42. [CrossRef]
65. Tochner, S.; Zohar, A.; Schmid, S. Route Hijacking and DoS in Off-Chain Networks. In Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFT), New York, NY, USA, 21–23 October 2020; ACM Digital Library: New York, NY, USA, 2020; pp. 228–240. [CrossRef]

66. Zhang, X.; Li, R.; Wang, Q.; Wang, Q.; Duan, S. Time-Manipulation Attack: Breaking Fairness against Proof of Authority Aura. In Proceedings of the 32nd ACM Web Conference (WWW), Austin, TX, USA, 30 April–4 May 2023; ACM Digital Library: New York, NY, USA, 2023; pp. 2076–2086. [CrossRef]
67. AlFaw, A.; Elmedany, W.; Sharif, M.S. Blockchain Vulnerabilities and Recent Security Challenges: A Review Paper. In Proceedings of the 3rd International Conference on Data Analytics for Business and Industry (ICDABI), Sakhir, Bahrain, 25–26 October 2022; ACM Digital Library: New York, NY, USA, 2022; pp. 780–786. [CrossRef]
68. Iqbal, M.; Matulevičius, R. Exploring Sybil and Double-Spending Risks in Blockchain Systems. *IEEE Access* **2021**, *9*, 76153–76177. [CrossRef]
69. Prashar, D. Analysis on Blockchain Vulnerabilities & Attacks on Wallet. In Proceedings of the 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 17–18 December 2021; pp. 1515–1521. [CrossRef]
70. Chen, X.; Wei, Z.; Jia, X.; Zheng, P.; Han, M.; Yang, X. Current Status and Prospects of Blockchain Security Standardization. In Proceedings of the IEEE 9th International Conference on Cyber Security and Cloud Computing (CSCloud)/IEEE 8th International Conference on Edge Computing and Scalable Cloud (EdgeCom), Xi'an, China, 25–27 June 2022; pp. 24–29. [CrossRef]
71. Leng, J.; Zhou, M.; Zhao, J.L.; Huang, Y.; Bian, Y. Blockchain Security: A Survey of Techniques and Research Directions. *IEEE Trans. Serv. Comput.* **2022**, *15*, 2490–2510. [CrossRef]
72. Sharma, S.; Shah, K. Exploring Security Threats on Blockchain Technology along with possible Remedies. In Proceedings of the IEEE 7th International conference for Convergence in Technology (I2CT), Mumbai, India, 7–9 April 2022; pp. 1–4. [CrossRef]
73. Islam, M.R.; Rahman, M.M.; Mahmud, M.; Rahman, M.A.; Mohamad, M.H.S.; Embong, A.H. A Review on Blockchain Security Issues and Challenges. In Proceedings of the IEEE 12th Control and System Graduate Research Colloquium (ICSGRC), Shah Alam, Malaysia, 7 August 2021; pp. 227–232. [CrossRef]
74. Bhutta, M.N.M.; Khwaja, A.A.; Nadeem, A.; Ahmad, H.F.; Khan, M.K.; Hanif, M.A.; Song, H.; Alshamari, M.; Cao, Y. A Survey on Blockchain Technology: Evolution, Architecture and Security. *IEEE Access* **2021**, *9*, 61048–61073. [CrossRef]
75. Hao, Y. Research of the 51% attack based on blockchain. In Proceedings of the 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA), Changchun, China, 20–22 May 2022; pp. 278–283. [CrossRef]
76. Wang, H.; Ge, C.; Liu, Z. On the Security of Permissionless Blockchain Systems: Challenges and Research Perspective. In Proceedings of the 4th IEEE Conference on Dependable and Secure Computing (DSC), Aizuwakamatsu, Japan, 30 January–2 February 2021; pp. 1–8. [CrossRef]
77. Snegireva, D.A. Review of Modern Vulnerabilities in Blockchain Systems. In Proceedings of the 6th International Conference on Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS), Yaroslavl, Russian, 6–10 September 2021; pp. 117–121. [CrossRef]
78. Ajayi, O.; Saadawi, T. Detecting Insider Attacks in Blockchain Networks. In Proceedings of the 8th International Symposium on Networks, Computers and Communications (ISNCC), Dubai, United Arab Emirates, 31 October–2 November 2021; pp. 1–7. [CrossRef]
79. Wang, S.; Yang, M.; Pearson, B.; Ge, T.; Fu, X.; Zhao, W. On Security of Proof-of-Policy (PoP) in the Execute-Order-Validate Blockchain Paradigm. In Proceedings of the 10th IEEE Conference on Communications and Network Security (CNS), Austin, TX, USA, 3–5 October 2022; pp. 317–325. [CrossRef]
80. Muralidhara, S.; Usha, B.A. Review of Blockchain Security and Privacy. In Proceedings of the 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 8–10 April 2021; pp. 526–533. [CrossRef]
81. Wang, S.; Yin, B.; Zhang, S.; Cheng, Y.; Cai, L.X.; Cao, X. A Selfish Attack on Chainweb Blockchain. In Proceedings of the 39th IEEE Global Communications Conference (GLOBECOM), Taipei, Taiwan, 7–11 December 2020; pp. 1–6. [CrossRef]
82. Zhang, P.; Zhou, M. Security and Trust in Blockchains: Architecture, Key Technologies, and Open Issues. *IEEE Trans. Comput. Soc. Syst.* **2020**, *7*, 790–801. [CrossRef]
83. Swambo, J.; Poinsot, A. Risk Framework for Bitcoin Custody Operation with the Revault Protocol. In Proceedings of the 25th International Workshops on Financial Cryptography and Data Security (FC), Virtual, 5 March 2021; Bernhard, M., Bracciali, A., Gudgeon, L., Haines, T., Klages-Mundt, A., Matsuo, S., Perez, D., Sala, M., Werner, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2021; pp. 3–20.
84. Kong, X.; Shen, M.; Che, Z.; Yu, C.; Zhu, L. Traffic Correlation for Deanonymizing Cryptocurrency Wallet Through Tor. In Proceedings of the 4th International Conference on Blockchain and Trustworthy Systems (BlockSys), Chengdu, China, 4–5 August 2022; Revised Selected Papers; Svetinovic, D., Zhang, Y., Luo, X., Huang, X., Chen, X., Eds.; Springer: Singapore, 2022; pp. 292–305.
85. Romiti, M.; Victor, F.; Moreno-Sanchez, P.; Nordholt, P.S.; Haslhofer, B.; Maffei, M. Cross-Layer Deanonymization Methods in the Lightning Protocol. In Proceedings of the 25th International Workshops on Financial Cryptography and Data Security (FC), Virtual, 5 March 2021; Borisov, N., Diaz, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2021; pp. 187–204.
86. Apostolaki, M.; Maire, C.; Vanbever, L. Perimeter: A Network-Layer Attack on the Anonymity of Cryptocurrencies. In Proceedings of the 25th International Workshops on Financial Cryptography and Data Security (FC), Virtual, 5 March 2021; Borisov, N., Diaz, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2021; pp. 147–166.

87. Ghesmati, S.; Fdhila, W.; Weippl, E. Studying Bitcoin Privacy Attacks and Their Impact on Bitcoin-Based Identity Methods. In Proceedings of the 18th Business Process Management (BPM): Blockchain and Robotic Process Automation (RPA) Forum, Rome, Italy, 6–10 September 2021; González Enríquez, J., Debois, S., Fettke, P., Plebani, P., van de Weerd, I., Weber, I., Eds.; Springer: Cham, Switzerland, 2021; pp. 85–101.
88. Wang, Q.; Yu, J.; Peng, Z.; Bui, V.C.; Chen, S.; Ding, Y.; Xiang, Y. Security Analysis on dBFT Protocol of NEO. In Proceedings of the 24th International Workshops on Financial Cryptography and Data Security (FC), Kota Kinabalu, Malaysia, 14 February 2020; Bonneau, J., Heninger, N., Eds.; Springer: Cham, Switzerland, 2020; pp. 20–31.
89. Wijaya, D.A.; Liu, J.K.; Steinfeld, R.; Liu, D. Transparency or Anonymity Leak: Monero Mining Pools Data Publication. In Proceedings of the 26th Australasian Conference on Information Security and Privacy (ACISP), Virtual, 1–3 December 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 433–450. [CrossRef]
90. Chen, H.; Pendleton, M.; Njilla, L.; Xu, S. A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses. *ACM Comput. Surv.* **2020**, *53*, 67. [CrossRef]
91. Yang, R.; Chang, X.; Mišić, J.; Mišić, V.; Zhu, H. Evaluating Fork after Withholding (FAW) Attack in Bitcoin. In Proceedings of the 19th ACM International Conference on Computing Frontiers (CF), Turin, Italy, 17–22 May 2022; ACM Digital Library: New York, NY, USA, 2022; pp. 67–74. [CrossRef]
92. Wang, Y.; Zuest, P.; Yao, Y.; Lu, Z.; Wattenhofer, R. Impact and User Perception of Sandwich Attacks in the DeFi Ecosystem. In Proceedings of the 40th CHI Conference on Human Factors in Computing Systems (CHI), New Orleans, LA, USA, 29 April–5 May 2022; ACM Digital Library: New York, NY, USA, 2022. [CrossRef]
93. Yang, Z.; Man, G.; Yue, S. Understanding Security Audits on Blockchain. In Proceedings of the 5th International Conference on Blockchain Technology and Applications (ICBTA), Xi'an, China, 16–18 December 2022; ACM Digital Library: New York, NY, USA, 2023; pp. 10–15. [CrossRef]
94. Harris, J.; Zohar, A. Flood & Loot: A Systemic Attack on The Lightning Network. In Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFT), New York, NY, USA, 21–23 October 2020; ACM Digital Library: New York, NY, USA, 2020; pp. 202–213. [CrossRef]
95. He, Z.; Li, J.; Wu, Z. Don't Trust, Verify: The Case of Slashing from a Popular Ethereum Explorer. In Proceedings of the 32nd ACM Web Conference (WWW) Companion, Austin, TX, USA, 30 April 2023–4 May 2023; ACM Digital Library: New York, NY, USA, 2023; pp. 1078–1084. [CrossRef]
96. Agarwal, R.; Thapliyal, T.; Shukla, S. Analyzing Malicious Activities and Detecting Adversarial Behavior in Cryptocurrency Based Permissionless Blockchains: An Ethereum Usecase. *Distrib. Ledger Technol.* **2022**, *1*, 8. [CrossRef]
97. Zaghloul, E.; Li, T.; Mutka, M.W.; Ren, J. Bitcoin and Blockchain: Security and Privacy. *IEEE Internet Things J.* **2020**, *7*, 10288–10313. [CrossRef]
98. Paavolainen, S.; Carr, C. Security Properties of Light Clients on the Ethereum Blockchain. *IEEE Access* **2020**, *8*, 124339–124358. [CrossRef]
99. Brotsis, S.; Kolokotronis, N.; Limniotis, K.; Bendiab, G.; Shiaeles, S. On the Security and Privacy of Hyperledger Fabric: Challenges and Open Issues. In Proceedings of the 16th IEEE World Congress on Services (SERVICES), Beijing, China, 18–23 October 2020; pp. 197–204. [CrossRef]
100. Bouichou, A.; Mezroui, S.; Oualkadi, A.E. An overview of Ethereum and Solidity vulnerabilities. In Proceedings of the 3rd International Symposium on Advanced Electrical and Communication Technologies (ISAECT), Marrakech, Marocco, 25–27 November 2020; pp. 1–7. [CrossRef]
101. Ferreira Torres, C.; Iannillo, A.K.; Gervais, A.; State, R. The Eye of Horus: Spotting and Analyzing Attacks on Ethereum Smart Contracts. In Proceedings of the 25th International Workshops on Financial Cryptography and Data Security (FC), Virtual, 5 March 2021; Borisov, N., Diaz, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2021; pp. 33–52.
102. Ashouri, M. An Extensive Security Analysis on Ethereum Smart Contracts. In Proceedings of the 17th EAI International Conference on Security and Privacy in Communication Networks (SecureComm), Virtual, 6–9 September 2021; Garcia-Alfaro, J., Li, S., Poovendran, R., Debar, H., Yung, M., Eds.; Springer: Cham, Switzerland, 2021; pp. 144–163.
103. Gupta, B.C.; Kumar, N.; Handa, A.; Shukla, S.K. An Insecurity Study of Ethereum Smart Contracts. In Proceedings of the 10th International Conference on Security, Privacy, and Applied Cryptography Engineering (SPACE), Kolkata, India, 17–21 December 2020; Batina, L., Picek, S., Mondal, M., Eds.; Springer: Cham, Switzerland, 2020; pp. 188–207.
104. Ji, M.; Liang, G.; Li, M.; Zhang, H.; He, J. Security Analysis of Blockchain Smart Contract: Taking Reentrancy Vulnerability as an Example. In Proceedings of the 7th International Conference on Advances in Artificial Intelligence and Security (ICAIS), Dublin, Ireland, 19–23 July 2021; Sun, X., Zhang, X., Xia, Z., Bertino, E., Eds.; Springer: Cham, Switzerland, 2021; pp. 492–501.
105. Chiu, W.Y.; Meng, W. Mind the Scraps: Attacking Blockchain Based on Selfdestruct. In Proceedings of the 26th Australasian Conference on Information Security and Privacy (ACISP), Virtual, 1–3 December 2021; Baek, J., Ruj, S., Eds.; Springer: Cham, Switzerland, 2021; pp. 451–469.
106. Ivanov, N.; Li, C.; Yan, Q.; Sun, Z.; Cao, Z.; Luo, X. Security Threat Mitigation for Smart Contracts: A Comprehensive Survey. *ACM Comput. Surv.* **2023**, *55*, 326. [CrossRef]
107. Maier, D.; Fäßler, F.; Seifert, J.P. Uncovering Smart Contract VM Bugs Via Differential Fuzzing. In Proceedings of the 5th Reversing and Offensive-Oriented Trends Symposium (ROOTS), Vienna, Austria, 18–19 November 2021; ACM Digital Library: New York, NY, USA, 2022; pp. 11–22. [CrossRef]

108. Wan, Z.; Xia, X.; Lo, D.; Chen, J.; Luo, X.; Yang, X. Smart Contract Security: A Practitioners' Perspective. In Proceedings of the 43rd International Conference on Software Engineering (ICSE), Madrid, Spain, 22–30 May 2021; pp. 1410–1422. [CrossRef]
109. Zhang, Z.; Lei, Y.; Yan, M.; Yu, Y.; Chen, J.; Wang, S.; Mao, X. Reentrancy Vulnerability Detection and Localization: A Deep Learning Based Two-Phase Approach. In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE), Rochester, MI, USA, 10–14 October 2022; ACM Digital Library: New York, NY, USA, 2023. [CrossRef]
110. Tjiam, K.; Wang, R.; Chen, H.; Liang, K. Your Smart Contracts Are Not Secure: Investigating Arbitrageurs and Oracle Manipulators in Ethereum. In Proceedings of the 3rd Workshop on Cyber-Security Arms Race (CYSARM), Virtual, 19 November 2021; ACM Digital Library: New York, NY, USA, 2021; pp. 25–35. [CrossRef]
111. Liu, Y.; Li, Y.; Lin, S.W.; Artho, C. Finding Permission Bugs in Smart Contracts with Role Mining. In Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), Virtual, 18–22 July 2022; ACM Digital Library: New York, NY, USA, 2022; pp. 716–727. [CrossRef]
112. Hwang, S.; Ryu, S. Gap between Theory and Practice: An Empirical Study of Security Patches in Solidity. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE), Seoul, Republic of Korea, 27 June–19 July 2020; ACM Digital Library: New York, NY, USA, 2020; pp. 542–553. [CrossRef]
113. Varun, M.; Palanisamy, B.; Sural, S. Mitigating Frontrunning Attacks in Ethereum. In Proceedings of the 4th ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI), Nagasaki, Japan, 30 May–3 June 2022; ACM Digital Library: New York, NY, USA, 2022; pp. 115–124. [CrossRef]
114. Samreen, N.F.; Alalfi, M.H. A Survey of Security Vulnerabilities in Ethereum Smart Contracts. In Proceedings of the 30th Annual International Conference on Computer Science and Software Engineering (CASCON), Toronto, ON, Canada, 10–13 November 2020; pp. 73–82.
115. Brent, L.; Grech, N.; Lagouvardos, S.; Scholz, B.; Smaragdakis, Y. Ethainter: A Smart Contract Security Analyzer for Composite Vulnerabilities. In Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), London, UK, 15–20 June 2020; ACM Digital Library: New York, NY, USA, 2020; pp. 454–469. [CrossRef]
116. Sayeed, S.; Marco-Gisbert, H.; Caira, T. Smart Contract: Attacks and Protections. *IEEE Access* **2020**, *8*, 24416–24427. [CrossRef]
117. Kushwaha, S.S.; Joshi, S.; Singh, D.; Kaur, M.; Lee, H.N. Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract. *IEEE Access* **2022**, *10*, 6605–6621. [CrossRef]
118. Pise, R.; Patil, S. A Deep Dive into Blockchain-based Smart Contract-specific Security Vulnerabilities. In Proceedings of the 1st IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS), Pune, India, 16–18 September 2022; pp. 1–6. [CrossRef]
119. Ji, S.; Kim, D.; Im, H. Evaluating Countermeasures for Verifying the Integrity of Ethereum Smart Contract Applications. *IEEE Access* **2021**, *9*, 90029–90042. [CrossRef]
120. Kushwaha, S.S.; Joshi, S.; Singh, D.; Kaur, M.; Lee, H.N. Ethereum Smart Contract Analysis Tools: A Systematic Review. *IEEE Access* **2022**, *10*, 57037–57062. [CrossRef]
121. Kissoon, Y.; Bekaroo, G. Detecting Vulnerabilities in Smart Contract within Blockchain: A Review and Comparative Analysis of Key Approaches. In Proceedings of the 3rd International Conference on Next Generation Computing Applications (NextComp), Flic-en-Flac, Mauritius, 6–8 October 2022; pp. 1–6. [CrossRef]
122. He, D.; Deng, Z.; Zhang, Y.; Chan, S.; Cheng, Y.; Guizani, N. Smart Contract Vulnerability Analysis and Security Audit. *IEEE Netw.* **2020**, *34*, 276–282. [CrossRef]
123. Sifra, E.M. Security Vulnerabilities and Countermeasures of Smart Contracts: A Survey. In Proceedings of the 4th IEEE International Conference on Blockchain (Blockchain), Espoo, Finland, 22–25 August 2022; pp. 512–515. [CrossRef]
124. Matulevicius, N.; Cordeiro, L.C. Verifying Security Vulnerabilities for Blockchain-based Smart Contracts. In Proceedings of the 11th Brazilian Symposium on Computing Systems Engineering (SBESC), Florianopolis, Brazil, 22–26 November 2021; pp. 1–8. [CrossRef]
125. Usman, T.A.; Selçuk, A.A.; Özarslan, S. An Analysis of Ethereum Smart Contract Vulnerabilities. In Proceedings of the 14th International Conference on Information Security and Cryptology (ISCTURKEY), Ankara, Turkey, 2–3 December 2021; pp. 99–104. [CrossRef]
126. Khan, Z.A.; Siami Namin, A. Ethereum Smart Contracts: Vulnerabilities and their Classifications. In Proceedings of the 8th IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; pp. 1–10. [CrossRef]
127. Staderini, M.; Palli, C.; Bondavalli, A. Classification of Ethereum Vulnerabilities and their Propagations. In Proceedings of the 2nd International Conference on Blockchain Computing and Applications (BCCA), Antalya, Turkey, 2–5 November 2020; pp. 44–51. [CrossRef]
128. Hajdu, A.; Ivaki, N.; Kocsis, I.; Klenik, A.; Gönczy, L.; Laranjeiro, N.; Madeira, H.; Pataricza, A. Using Fault Injection to Assess Blockchain Systems in Presence of Faulty Smart Contracts. *IEEE Access* **2020**, *8*, 190760–190783. [CrossRef]
129. Weber, K.; Schütz, A.E.; Fertig, T.; Müller, N.H. Exploiting the Human Factor: Social Engineering Attacks on Cryptocurrency Users. In Proceedings of the 7th HCI International Conference on Learning and Collaboration Technologies. Human and Technology Ecosystems (LCT), Copenhagen, Denmark, 19–24 July 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 650–668. [CrossRef]

130. Ivanov, N.; Lou, J.; Chen, T.; Li, J.; Yan, Q. Targeting the Weakest Link: Social Engineering Attacks in Ethereum Smart Contracts. In Proceedings of the 16th ACM Asia Conference on Computer and Communications Security (ASIA CCS), Virtual, 7–11 June 2021; ACM Digital Library: New York, NY, USA, 2021; pp. 787–801. [CrossRef]
131. Fröhlich, M.; Hulm, P.; Alt, F. Under Pressure. A User-Centered Threat Model for Cryptocurrency Owners. In Proceedings of the 4th International Conference on Blockchain Technology and Applications (ICBTA), Xi'an, China, 17–19 December 2021; ACM Digital Library: New York, NY, USA, 2022; pp. 39–50. [CrossRef]
132. Fröhlich, M.; Gutjahr, F.; Alt, F. Don't Lose Your Coin! Investigating Security Practices of Cryptocurrency Users. In Proceedings of the 15th ACM Designing Interactive Systems Conference (DIS), Eindhoven, The Netherlands, 6–10 July 2020; ACM Digital Library: New York, NY, USA, 2020; pp. 1751–1763. [CrossRef]
133. Buja, A.G.; Katan, M.; Nasrijal, N.M.H.; Alwi, S.F.S.; Siang, T.G. Into the Look: Security Issues, Crypto-Hygiene, and Future Direction of Blockchain and Cryptocurrency for Beginners in Malaysia. In Proceedings of the 6th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Kedah, Malaysia, 1–3 December 2021; pp. 1–6. [CrossRef]
134. OpenZeppelin. Math. 2023. Available online: https://docs.openzeppelin.com/contracts/2.x/api/math (accessed on 19 October 2023).
135. Solidity. Solidity v0.5.0 Breaking Changes. 2018. Available online: https://docs.soliditylang.org/en/latest/050-breaking-changes.html (accessed on 19 October 2023).
136. Solidity. Security Considerations. 2023. Available online: https://docs.soliditylang.org/en/latest/security-considerations.html (accessed on 19 October 2023).
137. Ethereum Foundation. ERC-4337: Account Abstraction. 2023. Available online: https://www.erc4337.io (accessed on 19 October 2023).
138. Ethereum Foundation. EIP-608: Hardfork Meta: Tangerine Whistle. 2017. Available online: https://eips.ethereum.org/EIPS/eip-608 (accessed on 19 October 2023).
139. Ethereum Foundation. EIP-150: Gas Cost Changes for IO-Heavy Operations. 2016. Available online: https://eips.ethereum.org/EIPS/eip-150 (accessed on 19 October 2023).
140. Ethereum Foundation. Ethash. 2023. Available online: https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/mining-algorithms/ethash/ (accessed on 19 October 2023).
141. Ethereum Foundation. EIP-155: Simple Replay Attack Protection. 2016. Available online: https://eips.ethereum.org/EIPS/eip-155 (accessed on 19 October 2023).
142. Ethereum Foundation. EIP-161: State Trie Clearing (Invariant-Preserving Alternative). 2016. Available online: https://eips.ethereum.org/EIPS/eip-161 (accessed on 19 October 2023).
143. Ethereum Foundation. Consensus Mechanisms. 2023. Available online: https://ethereum.org/en/developers/docs/consensus-mechanisms/ (accessed on 19 October 2023).
144. Hyperledger Indy. 2023. Welcome to Indy Plenum's Documentation! Available online: https://hyperledger-indy.readthedocs.io/projects/plenum/en/latest/index.html (accessed on 19 October 2023).
145. Yang, G.; Lee, K.; Lee, K.; Yoo, Y.; Lee, H.; Yoo, C. Resource Analysis of Blockchain Consensus Algorithms in Hyperledger Fabric. *IEEE Access* **2022**, *10*, 74902–74920. [CrossRef]
146. Schäfer, J.; Malinka, K.; Hanácek, P. Peer-to-Peer Networks Security. In Proceedings of the 3rd International Conference on Internet Monitoring and Protection (ICIMP), Bucharest, Romania, 29 June–5 July 2008; ACM Digital Library: New York, NY, USA, 2008; pp. 74–79. [CrossRef]
147. Graux, H. Whose Data is It Anyway? Diverging Perspectives in EU Policy on the Current and Future Role of the Citizen in Digital Government. In Proceedings of the 24th Annual International Conference on Digital Government Research (DG.O), Gdansk, Poland, 11–14 July 2023; ACM Digital Library: New York, NY, USA, 2023; pp. 508–513. [CrossRef]