Security Aspects of Cryptocurrency Wallets—A Systematic Literature Review

SABINE HOUY, Umeå University, Sweden
PHILIPP SCHMID, Technical University of Munich, Germany
ALEXANDRE BARTEL, Umeå University, Sweden

Cryptocurrencies are gaining prominence among individuals and companies alike, resulting in the growing adoption of so-called cryptocurrency wallet applications, as these simplify transactions. These wallets are available in a myriad of different forms and specifications. All of them are susceptible to various ways the attacker can exploit the vulnerabilities and steal money from victims. Cryptocurrency wallets create a unique field as they combine features of password managers, banking applications, and the need to keep their users and their transactions anonymous. We collect the findings from previous literature to provide an overview of the different attack surfaces, possible countermeasures, and further research. Existing literature focused on one of the features mentioned before, while we considered all of them. Our systematic study shows that there is a considerable variety of attack vectors, which we have divided into six subcategories, (i) Memory and Storage, (ii) Operating Systems, (iii) Software Layer, (iv) Network Layer, (v) Blockchain Protocol, and (vi) Others. We have found a large gap between the possible countermeasures and their actual adoption. Therefore, we provide a list of possible directions for future research to tackle this gap.

CCS Concepts: • **Security and privacy** → *Software and application security;*

Additional Key Words and Phrases: Blockchain, wallet, cryptocurrency, bitcoin, vulnerability

ACM Reference format:
Sabine Houy, Philipp Schmid, and Alexandre Bartel. 2023. Security Aspects of Cryptocurrency Wallets—A Systematic Literature Review. *ACM Comput. Surv.* 56, 1, Article 4 (August 2023), 31 pages.
https://doi.org/10.1145/3596906

1 INTRODUCTION

In today's world, cryptocurrency is becoming increasingly popular, even with personal use, which brings an upward trend in using wallet applications [1]. These wallets are the primary means used to manage the cryptocurrency. Such wallets attract a considerable attention from attackers. The wide variety of available types and specifications of wallets makes it considerably difficult for the user to select a safe and suitable one for use. This is mainly due to lack of proper overview of the applications' various vulnerabilities, which could help with this decision, since a vulnerability can result in the loss of part or all of the monetary value of the tokens. When media discussed attacks on wallets and the resulting loss of assets, it is usually not a single user who is affected but a large number of users who use the same exchange service. The the attacks' impacts are presented

Authors' addresses: S. Houy and A. Bartel, Umeå University, Universitetstorget 4, Umeå, Västerbotten, Sweden, 901 87; emails: sabine.houy@umu.se, alexandre.bartel@cs.um.se; P. Schmid, Technical University of Munich, Arcisstraß 21, Munich, Bavaria, Germany; email: philipp.schmid@tum.de.

0360-0300/2023/08-ART4 $15.00
https://doi.org/10.1145/3596906

to the public by the media at the level of the respective exchange service and not the individual user. One of the most famous and biggest heist targeted an exchange called Mt.Gox where an esteemed 740,000 Bitcoins were stolen [67]. However, not only was Mt.Gox affected but also NiceHash [67], YouBit [67], and Coincheck [29]. These exchanges will stay in the focus of attacks, since they can lead to highly profitable thefts [39].

In 2015, Bonneau et al. [25] provided a comprehensive overview of the challenges faced by Bitcoin and other cryptocurrencies. They give a detailed survey of what has been conducted so far in this area, "ranging from novel payment protocols to user-friendly key management" [25]. They discuss the vulnerabilities and defenses. However, they only considered wallet applications in the context of key management. Our work supplements Bonneau et al.'s research by summarizing today's state of wallet-related vulnerabilities and countermeasures. Furthermore, our work aims to provide more detailed insights into security- and privacy-related challenges such as deanonymization attacks and new approaches regarding key recovery. As cryptocurrency wallets offer a more complex and layered structure than other applications only dealing with credentials and key management, such as password managers, they leave more opportunities for attackers. Besides storing login credentials, crypto wallets also function as banking applications. Another critical feature of crypto wallets that attackers can target is that they try to keep their users anonymous. Therefore, looking at cryptocurrency wallets separately is necessary to cover all possible weaknesses. We achieve this by undertaking a systematic literature review to give a structured overview of identified vulnerabilities of cryptocurrency wallets to provide a taxonomy of cryptocurrency attacks and defenses. We outline existing countermeasures that developers, for example, could utilize in their future implementations. The following research questions will be answered in this article:

RQ1: What are the most common vulnerabilities of cryptocurrency wallet applications?
RQ2: What are the existing countermeasures to mitigate the known vulnerabilities?

Our research focuses on different vulnerabilities and resulting attacks targeting users of cryptocurrency wallet and associated countermeasures. The attacks described exploit weaknesses ranging from flaws specific to a particular wallet application to general issues in the blockchain. The defenses cover the design of new applications by remodeling the underlying architecture and structure of the wallets, and smaller adjustments in existing approaches, designs, and protocols. It summarizes the most common vulnerabilities of cryptocurrency wallet applications and how to best avoid or mitigate them. It focuses on the wallets themselves, and it only mentions some of the basic principles of the typical attacks on the Blockchain protocol. It does not address implementation-specific problems on the Blockchain protocol, for example, cryptographic functions used in Bitcoin, Ethereum, or other protocols, as these form a separate field of research and are beyond the scope of this article.

By providing the most common attacks, we aim to raise the awareness among cryptocurrency stakeholders, including the wallet's owner, the developers of the cryptocurrency application, and the exchanges of cryptocurrencies. For this reason, it is important to provide a broad overview of the current vulnerabilities and countermeasures.

We start by explaining the technical background in Section 2, including a general description of Blockchain, cryptocurrency wallets, and wallet types. Section 3 summarizes our results, including the wallet vulnerabilities described or identified in the previously selected papers as well as the determined countermeasures. In Section 4, we propose future research directions. Finally, we summarize the most important conclusions and highlight our main findings in Section 5. The methods used to collect the relevant papers can be found in the appendix, including database lookups to extract the relevant papers using search queries containing different combinations of keywords

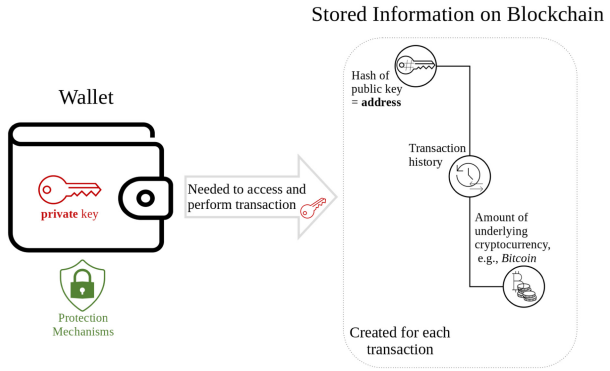


Fig. 1. Simplified explanation of role of wallets within cryptocurrency transaction.

and backward-forward reference checking. The inclusion and exclusion criteria are defined and the outcome of the selection process is shown.

2 BACKGROUND

Blockchain technology was first proposed in 1991 [96], and the first currency using Blockchain, Bitcoin, was published in 2008 by Satoshi Nakamoto [70]. The Blockchain is an immutable transaction log and uses consensus algorithms to bring every node in the network on the same status. Information is stored by creating transactions (*TX*) and broadcasting them via a block throughout the network. The nodes check the *TX*, and if they are valid, then they will be included in blocks together with other information as a nonce and the (cryptographic) hash of the previous block, the signature of the coins' owner. The block is valid only when all the including *TX* are valid and they are accepted by majority of nodes within the network. All *TX* are valid if all the signatures fit their owners' public addresses and the amount of transferred coins is meaningful. Additionally, the input has to be higher or equal to the output, no blacklisted addresses are included, and all other restrictions need to be matched [34]. The public address is a unique identifier that is used similarly to a bank account number. The generation of it depends on the cryptocurrency and can be based on the public key. The precise requirements for the information enclosed within a *TX* depends on the implementation of the currency. Figure 1 provides a simplified illustration of the main components needed for a transaction using a cryptocurrency wallet. The wallet itself does not store any coins. However, it stores and protects the private key, which is essential to perform transactions and thus to use the coins. Other information as the hash of the public key, which often equals the address, the transaction history, and the amount of cryptocoins, is stored on the Blockchain. One way to protect the private key and other sensitive data stored in the wallet is encryption. Such encryption is achieved using a locking mechanism, e.g., a PIN or password could be required to access the application (see Figure 2).

PINs and other protection mechanisms as passwords and biometric authentication (e.g., fingerprints) help prevent the access of others and thus unauthorized transactions. This means encryption in terms of storage. The stored data are encrypted and thus cannot be viewed by third parties. To use the application, a password, PIN, or similar is required to decrypt the data so that the application can access it. Figure 2 gives a brief overview of some protection techniques. If, for example, the PIN is found in the memory, then it can be used to gain full access to the wallet and perform transactions.

Cryptocurrency wallets are categorized based on how they access the network, the medium where the wallet is stored, whether it stores the whole chain or just parts of it, and whether keys

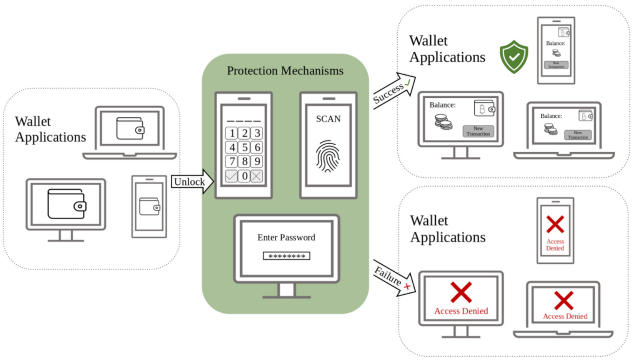


Fig. 2. Protection mechanisms to access wallet applications.

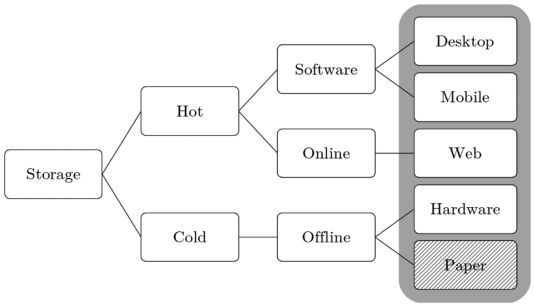


Fig. 3. Types of wallets [86].

The wallet types shaded in gray are discussed in this article. The *Paper* wallet is hashed, because it is not a technological implementation of a wallet and therefore not relevant for this study.

are stored locally or on a (remote) server. The following paragraphs explain the different types of wallets in detail. Figure 3 gives a brief overview of the different wallet types. The first distinction between wallets is whether it is a *hot* or *cold* wallet [86]. Hot wallets are the ones that have (or require) a consistent connection with the Internet. They are usually in the form of a program or application installed or accessed on a smartphone or computer. People can use them either as online or web wallets, which are browser applications that do not need to be downloaded and installed, or in the form of software wallets, which are further categorized between desktop and mobile applications. The former are programs installed and used on a computer, and the latter are mobile apps [86]. Cold wallets or offline wallets do not have access to a network or the Internet. Therefore, they are conceived as a special device that is used only for this purpose. This device is also called a hardware wallet. Another type of cold wallet would be a paper wallet, which means writing down the necessary information as the private key. However, we will not focus on paper wallets, since they are non-technological implementations but literally just a piece of paper, which is highly impractical to use. In general, it is assumed that cold wallets are more secure, since they do not connect to the Internet [86, 96]. In the following section, the specific application types shaded in gray in Figure 3 are explained in more detail.

- *Desktop* wallets are software that is downloaded and installed on a computer. They usually give complete control of the keys and funds to the user. In addition, sensitive data such as the keys are stored in local files on the device and are typically encrypted. Therefore, it is

crucial to backup the wallets in case of loss or damage. Most applications offer the option of recovery using a seed phrase [86].

- *Mobile* wallets are similar to desktop wallets. They are installed in the form of apps on a mobile device and give key management to the user. It is essential to regularly backup the application, since mobile breakdowns, theft or loss, and security breaches are common. The backup is stored on the device or transferred to a cloud server depending on the used app.
- *Web* wallets are often considered the least secure type of wallets [86] as they are owned by third-party providers responsible for critical storage. However, a usability advantage is that they do not require installation and can be accessed from anywhere. As a result, the wallets cannot be lost and can be recovered easily, since even if the phone is lost over which the web application was accessed, the owner can still access it from another device. However, if the owner loses or forgets the login credentials, then all assets stored in the wallet are inaccessible and thus lost.
- *Hardware* wallets are known to be the most secure as they have no connection to the Internet. However, there is no way to restore the wallet unless there is an additional backup wallet in case of loss. The key management is entirely in the hands of the user. This kind of wallet is often combined with software or online wallets to perform the transactions.

In terms of security, one can distinguish three different levels: (i) *full-service*, (ii) *signing-only*, and (iii) *distributing-only* wallets. The first type offers all services at once: It can generate keys, sign, and perform transactions. Full-service wallets need to have Internet access, which makes them also more prone to network attacks. The second one, signing-only wallets, tries to reduce those vulnerabilities. They are not connected to the Internet and therefore cannot perform transactions, which means users must combine them with an online wallet. The last type is distribution-only wallets, which aim to minimize the problems of full-service wallets by being organized in a network of wallets. These wallets are only responsible for distributing the public keys and are used to generate **hierarchically distributed (HD)** wallets from a master key [96].

Seed passphrases based on mnemonics are often used for recovery, as they are easier to remember than conventional passwords. For this purpose, a seed of usually 12–24 words is generated, where each is a different word, predefined or self-selected. The key is then generated based on these words. The users only have to remember the words and their sequence correctly, and then they can recalculate the key repeatedly and recover the wallet [100].

3 RESULTS OF SYSTEMATIC LITERATURE REVIEW

This section provides a summary of the results regarding existing attacks and countermeasures introduced by the reviewed literature.

3.1 Attacks

We classified the existing attacks into six categories based on different attack vectors resulting from our literature research:

(a) Memory and Storage
(b) Operating System
(c) Software Layer
(d) Network Layer
(e) Blockchain Protocol
(f) Others

The attacks assigned to the first category focus on hardware-related weaknesses (see (a) Memory and Storage); the second category contains all attacks that are specific to a particular OS (see

(b) Operating System). The third category covers all attacks that exploit vulnerabilities in the implementations of the wallet applications (see (c) Software Layer); the fourth category includes attacks targeting weaknesses unique to the networks of various cryptocurrencies (see (d) Network Layer). The fifth category contains attacks on the Blockchain protocol (see (e) Blockchain Protocol), and the last category includes all the attacks that do not fit in the other categories (see (f) Others). Each category is further divided into attacker models based on the requirements, and assumptions and prerequisites to perform the different attacks described in the literature. These attacker models can have different goals or all the same one. We summarized all attacker models and their goals in Table 1.

(a) Memory and Storage. The literature assigned to this category is divided into three different attacker models. The first model has full physical access to the device but no other knowledge. This includes the access to hardware components such as the RAM and disk. Moreover, the attacker is able to access the data stored on the hardware components from the OS, and thus the device needs to be turned on and unlocked. The second attacker model contains all the theoretical attacks. For example, files that are needed to perform the attack are self-generated in the literature. However, to perform the same attacks in reality, access to these files would be required. This could be achieved by full physical access as described in the first attacker model or, for instance, by installing malware that can access the hardware. The third attacker model has unlimited access to the login interface meaning, that they have an unlimited amount of login attempts. All attacker models have the same goal, they want to get access to the user wallets to steal their coins by retrieving login credentials and other useful information such as the private key.

(1) *Attacker Model 1—physical access:* Attacks on the memory level usually focus on extracting data used in a subsequent attack performed on a different level. The data obtained by a RAM analysis can include the wallet's private keys, seeds, PIN codes, and the transaction history [45, 90]. In case of web application, Zollner et al. [100] showed that it is possible to extract the URLs of all used wallets and artifacts such as the mnemonic passphrase and the wallet ID.
Koerhuis et al. [60] conducted an artifact analysis of two privacy-oriented cryptocurrencies, *Monero* and *Verge*. They examine the software running on a computer used to execute transactions. They performed a memory and disk analysis. It was possible to extract the wallet passphrase, the entire transaction history, and the public and stealthy addresses for Monero and Verge.

(2) *Attacker Model 2—theoretical attacks:* Praitheeshan et al. [75] performed two attacks focusing on the keystore file of Ethereum wallets. This specific file stores the account credentials of the wallets. A key that is at least eight characters long protects this file. If attackers can crack this key (password), then they gain full control of the wallet and thus the Ethereum address. The keystore file was created in the experimental setup, and various eight character long passwords were generated. First, they performed a brute-force attack using Hashcat [3]. Hashcat could not crack one of the passwords in the predefined time limit of one hour. The estimated time was ten years. Using the mask option made it possible to crack passwords consisting of a sequence of numbers. The mask option can only be used when more detailed information about the password is available, e.g., only numbers were used. The second attack run against the generated test files was a dictionary attack using the rockyou [8] dictionary and Hashcat. Hashcat was able to crack "weak passwords with single characters, continues digits or letters, combination of simple digits and letters and common words" [75] in less than an hour.
Not only implementation and structure misconfigurations lead to incorrect key storage but also the unfamiliarity of the users. Often they do not know what function the private key has

Table 1. Overview of Attacker Models

Section	Attacker Model	Description	Goals	Literature
(a) Memory and Storage	a-1	• Full physical access • Access to hardware components • Device turned on and unlocked	Steal coins	[45], [90], [100], [60]
	a-2	• Unlimited login attempts		[75], [36]
	a-3	• Theoretical attack • Self-generated files • In reality access to these files		[94]
(b) Operating Systems	b-1	• USB debugging • Physical access to rooted device	Steal coins	[63], [46], [45], [95], [69], [90]
	b-2	• Malicious application installed • High privileges/ privilege escalation		[63], [62], [90], [58], [91]
(c) Software Layer	c-1	• Presence of underlying implementation flaws • Access to specific files (e.g., log files) • Physical or non-physical (e.g., posted online)	Steal coins	[100], [45], [92], [26], [83], [59]
	c-2	• Impersonate server âĘŠ prevent benign server from starting • Impersonate user		[27]
	c-3	• Faulty use of libraries • Or use of faulty libraries		[89], [50]
	c-4	• Sending fake transactions		[55]
(d) Network Layer	d-1	• Passive access to network âĘŠ monitor • Network analysis	Deanonymize (Steal coins)	[22], [23], [88], [20], [80], [68], [42], [21], [13], [56], [12]
	d-2	• Active access to network âĘŠ modify • Inject software/program		[88], [61], [96], [31], [14], [71]
(e) Blockchain Protocol	e-1	• Acceptance of unconfirmed TX • Accept before validation	Get coins (Steal coins)	[96], [31], [54], [74], [53], [30], [73], [93], [85], [71], [41], [34], [51], [97], [57], [41], [66], [77], [16]
	e-2	More than 50% of computing power of the network		[96], [93], [85], [71]
	e-3	Enough computing power to perform DoS		[96], [31]
	e-4	• Network allows votes per IP and not per node • Attacker owns multiple IP addresses		[96], [31], [35], [85**?**], [71]
	e-5	• Own multiple nodes in network • Blackout occurs or ISP goes down		[96], [31], [47], [85], [71]
(f) Others	f-1	Exchange platform or online wallet provider: • Is the attacker • Is vulnerable to attacks	Deanonymize Steal coins	[22]
	f-2	Faulty implementation at any level		[15], [44], [37]
	f-3	*Vulnerable air-gapped wallets*		[33], [43]

or that it exists at all. The information provided by the wallets in this regard is inadequate or non-existent, leading to incorrect use on the user side [36].

(3) *Attacker Model 3—unlimited login attempts:* With the increase of users of Blockchain technology, more people have to handle their private keys. *Brain wallets* were introduced to simplify the storage of keys. Brain wallets are Blockchain wallets derived from a user-given or randomly generated passphrase, mainly containing 12 or 24 words. This method replaces the need to have a physical device for storing the key on it and makes it impossible for attackers

to steal it, since it is not permanently saved (except in the user's memory). Nevertheless, offering access to it via a passphrase exposes a weakness, because an attacker can guess the password without any restrictions, and if guessed right, then the attacker has complete control over the wallet. The study of Vasek et al. [94] revealed that most brain wallets were drained within 24 hours or less.

The most critical part of memory analysis is the improper storage of sensitive information such as private keys or seeds. To gain access to this information, physical access to the device on which the wallet application was used is required. Also, the device must be turned on. However, it is not possible to extract all the data stored in memory to achieve this, user access (for most web applications), or root access (for most app applications) is required. These cases are covered in the following parts.

(b) Operating System. When it comes to operating system vulnerabilities, most papers focus on Android applications, since it has the biggest market share of mobile operating systems [7]. Therefore, most of the vulnerabilities mentioned in this section are specific to Android.

The attacker model always assumes that the attacker has gained access to the device. Two attacker models are considered: The attacker gained access using USB debugging, or the attacker installed a malicious application on the victim's device. When using USB debugging, physical access is needed and the device needs to be rooted. Both attacker models have the same goal, they want to gain access to sensitive information such as login credentials by that they can then steal coins.

(1) *Attacker Model 1—USB debugging:* The attacker can perform an artifact analysis, e.g., scan the mobile device's memory and search for sensitive information [63]. He et al. [46] showed that extracting the PIN code from the Huobi Wallet [4] and the imToken [6] application is possible.
Moreover, Haigh et al. [45] analyzed eight different Android applications installed on a rooted device. They acquired sensitive data during an artifact analysis using **Android Debugging Bridge (adb)**. Volety et al. [95] performed a brute-force attack based on data found during memory scans that were possible after gaining access due to the debugging mode. Angelica Montanez [69] performed an artifact analysis of wallet applications installed on Android and iOS devices. It was possible to extract information regarding the active presence of a wallet app for both operating systems; gaining data about a past application was only possible for Android apps. For the Android device, the adb pull command was particularly successful in extracting valuable data of installed and active applications as well as past ones.
Uddin et al. [90] performed an extensive analysis of 311 Android wallet applications, revealing that 111 apps store key-related information in plain. Moreover, only 20 apps implement their own keyboard, which can prevent the prediction of the mnemonic passphrase by using the user dictionary, which stores the words typed into the default keyboard. Only 70 apps check whether the device they are running on is root. A rooted device can lead to security risks as monitoring activities or sending sensitive information to an external server in the presence of an installed malicious application, which is also interesting for the second attacker model.
(2) *Attacker Model 2—malicious application installed:* A vulnerability that is generally Android specific, but can be used as an attack against crypto wallets, exploits the so-called Accessibility Mode in Android. The permission `BIND_ACCESSIBILITY_SERVICE` can be requested by any application and enables, if set to true (=1), the Android accessibility features [62, 63]. Those features allow applications to access all GUI events, including all displayed information on the screen [62] except the password textbox, since this one is protected by

default [63]. Leguesse et al. [62] even announced that it would be possible to "*potentially backdooring Android's security model*" [62]. An attacker can use this vulnerability by first uploading a benign application to the PlayStore with permission to use the accessibility mode. The `BIND_ACCESSIBILITY_SERVICE` permission is not classified as suspicious or dangerous by Google [62]. Second, when opened, the installed application needs to trick the user into enabling the setting to allow installation from third-party sources. This can be achieved by creating an overlay so that users do not realize which button they click. After that, the malicious application needs to create another overlay tricking the user into downloading an additional malicious app via a hidden link from some third-party holding permissions marked as malicious by the PlayStore [63]. Moreover, Li et al. [63] explain how an application with enabled accessibility features could capture sensitive information when combined with a third-party keyboard. The password text field cannot be observed easily, but the attacker could listen to the click events triggered by the third-party keyboard.

Once the malicious application is installed on the victim's device, the attacker has even more options than those mentioned above. Android divides its storage into external and internal storage, mainly differentiated by their security levels [63]. Each app has its internal storage in which it stores all its app-specific data. Only the app itself or root can access it. The permission to access the external storage containing photos, documents, videos, downloads, and so on, can be requested by any application during runtime. To force the user to grant this permission, the app could show a pop-up window that needs to be accepted to use the application. Another option would be to mimic an older Android version, since no runtime request is required for those. Backup files are usually stored in a local file or external cloud. Assuming the first in combination with access permission to the external storage, an attacker's app could read the backup file, if not encrypted, and thus gain access to sensitive information such as transaction ids. Regularly, the field `android:allowBackup` is set to false in the Manifest.xml by default [63, 90].

When an application is holding the `SYSTEM_ALERT_WINDOW` permission, it can automatically draw any floating window and thus tamper the user interface by, e.g., capturing the password or manipulating the entered transaction credentials [63]. Another way to change the transaction data is to modify the clipboard, especially the receiver's address, since it is often too long and complicated to memorize. Hence, users copy it to the clipboard and then paste it into the wallet app. And precisely between these two actions, the attacker can intervene and replace the copied address with their own or any other arbitrary address without the victim's notice. This weakness is independent of the application and even the used operating system [58, 63]. Moreover, Ulqinaku et al. [91] presented an attack exploiting the scan-and-pay mechanism by creating an overlay containing a fake QR code.

(c) Software Layer. Some issues arise neither on the memory or the operating system level but on the level of the used software. All attacks described in this section are based on implementation flaws either directly or indirectly, for instance, by using a library containing an implementation flaw. All attackers have the goal to exploit these flaws in one way or another to steal coins. The attackers are divided into four different models. The first attacker model has access to specific data by either having physical access to the device or by having access to files containing this data such as log files that have been posted online by users. The second attacker model tries to impersonate the user either directly if the wallet is using Remote Procedure Calls or indirectly by first impersonating the server. The last approach requires the attacker to be able to prevent the benign server from starting. The third models exploits the use of vulnerable libraries and the last one sends messages containing fake transactions to the users. Therefore, it is required to have the ability to contact the users in some way.

(1) *Attacker Model 1—physical access or access to specific file:* Cryptocurrency wallets leave artifacts on the device, even after removing the wallet [100]. It is possible to find artifacts in the browser history, temporary storage, and cookies. Some cryptocurrency wallets save their credentials in an inappropriate file format, meaning in a non-encrypted form. Keeping the private keys in an unencrypted format can lead to theft of the wallet's complete content. Saving the transactions in plaintext can be a privacy issue, since it can connect an address to a person [45]. The same applies to contact lists that are saved in plaintext [92]. Some users are sharing debugging logs of the wallets on publicly available platforms (e.g., Pastebin). Some of those logs contain sensitive information like the private keys or information that enables to infer the private key [26]. Other wallets showed deficiencies by using a hardcoded encryption key or other sensitive information [45, 83].
The key generation process is one of the most critical steps in using cryptocurrency and needs to be done securely to protect the user from loss or theft. The nonce plays an essential role in the creation process and needs to be kept secret, and in practice, it is unlikely that the same nonce is picked more than once, since the possible set of nonces is roughly about 2^{256} possibilities [26]. Nevertheless, the reuse of nonce already occurred within the Bitcoin Blockchain, and it is most likely that it happened due to a weak random number generator, a bad implementation of a wallet, or a reset of a virtual machine and thus to reset to the same seed [26, 59].

(2) *Attacker Model 2—user and server impersonation:* Many desktop cryptocurrency wallet applications use open remote procedure calls to extend the wallet's functionality with other Blockchain-based applications, e.g., web-browser extensions [27]. By offering this **Remote Procedure Call (RPC)** interface, it is possible to impersonate the wallet owner in some cases without the need for a password. By impersonating the wallet owner, it is possible to start a transaction to an address owned by a malicious actor. Some other wallets enable the function to use basic access authentication, but Bui et al. [27] found a way to get access to the wallet by impersonating the server first. They started a server on the same port as the regular server and prevented it from starting. After the user sends its credentials to the malicious server, it shuts down and waits until the regular server restarts. After the restart, an impersonation as the wallet owner can be achieved. There are cryptocurrency wallet applications that use digest access authentication. Bui et al. [27] managed to impersonate the server in the same way as in the previous example. The attacker can capture all the commands sent to the server, especially the wallet generation process, and thus can own a copy of the new keys. The difference from the previous example is that they consider the client impersonation as not practical, because they do not receive the credentials in plaintext but as an MD5 hash that has to be cracked before.

(3) *Attacker Model 3—vulnerable libraries:* Another potential source of danger is the faulty use of libraries. Tschannen et al. [89] analyzed the usability of the library *libbitcoin* "well-known C++ implementation of the bitcoin system." The inappropriate use of it can lead to bugs and security vulnerabilities, including zero-day attacks. Their evaluation of libbitcoin showed that, e.g., misleading documentation and function names lead to the wrong usage in source code of applications and thus to security issues. Hu et al. [50] presented a vulnerability of the *BitcoinJ* client library, which is used for communicating with the **Simple Payment Verification (SPV)** client. Wallets as Bitcoin Wallet and Mycelium utilize this library. The SPV client establishes a connection to a **Full Node Client (FNC)** via TCP. However, if this connection is torn down, then the SPV client will automatically establish a new connection to an available FNC, which can be interfered with by a malicious third party. Thus, the attacker can extract the wallet's Bitcoin address. Moreover, the transactions are downloaded in the

background without the user's knowledge. Therefore, if a man-in-the-middle attack occurs, then the attacker can add or change transactions. Furthermore, Hu et al. found that some Bitcoin wallets, e.g., Bitcoin Wallet and Coinbase, violate the decentralization property of the network as they do not allow the wallet to join the network directly but through the wallet's provider's server.

(4) *Attacker Model 4—sending fake transactions:* Other implementation mistakes can lead to *Bait and Switch* vulnerabilities allowing users to steal coins after the faulty new source code was pushed. An attack called *Fabricated Transaction* means that fake transactions are sent to users. Kaushal et al. [55] analyzed multiple wallet applications and found that only one is secure against the attacks mentioned earlier, the Bitcoin Core wallet.

(d) Network Layer. On the network layer, we distinguish between two attacker models. The first model is a passive attacker who has access to the network and can see what all other user can see. It is possible that the attacker has additional information such as data from social media platforms. This model also contains attacks used by law enforcement. The second attacker model, contains an active attacker, which means they need the ability to modify the network.

(1) *Attacker Model 1—passive access to network:* It is possible to deanonymize users using different techniques. One of those techniques is called clustering. Biryukov and Tikhomirov [22, 23] deployed this technique, and by analyzing wallet applications and their underlying protocols, they deanonymized users. In addition to their clustering, they also tried to estimate the issuing user's IP, giving insightful information for an attacker. Their approach analyzed the propagation time between each of the nodes. They successfully clustered smaller networks such as the Bitcoin test network and Zcash but had problems with more extensive networks (e.g., Bitcoin mainnet), because clustering these networks requires significantly more computing power, and their computing resources were insufficient for this.
Additionally, Teomete et al. [88] and Bergman et al. [20] provided overviews of deanonymization methods for Bitcoin that can be utilized by law enforcement. Teomete et al. listed three techniques based on clustering. The first approach, developed by Reid et al. [80], takes advantage of the Blockchain's property that when Bitcoins are transferred, the link between the sender and the receiver is publicly accessible through blocks in the Blockchain. A transaction graph can be created between the various identities based on these links, even if different addresses are generated for each transaction. The identities in the created network can be grouped into the same entities. It is possible to identify the most commonly used transactions using these groups, and if an identity is revealed in between, then any Bitcoin address associated with that identity can be tracked using financial and customer transaction histories. The second method, summarized by Teomete et al. and developed by Meiklejohn et al. [68], assumes that when Bitcoins from two different addresses are combined, those bitcoins, and therefore the addresses, belong to the same user. The last technique utilizes website analytic cookies to locate Bitcoins transactions by cross-matching information and data on the Blockchain [42]. Some clustering methods with published heuristics were tested by Neudecker et al. [72]. They showed that only a small share of clusters could be assigned to a single IP address. However, adjusted heuristics and a preciser network observation could improve the results. Biryukov et al. [21] introduced a new method to deanonymize clients in the Bitcoin network based on the entry nodes and the established outgoing connections. Androulaki et al. [13] conducted a study mimicking Bitcoin as the primary payment method used at a university. Their evaluation showed that using the security features recommended by Bitcoin, the identity of 40% of the users can be revealed.

Khalilov et al. [56] performed an extensive survey regarding privacy issues in Bitcoin and similar systems. They concluded that for Bitcoin, improvement is needed. As mentioned above, many attack methods exist, most of which result from the transparency that all transactions are broadcasted in the network. They also stated that combining data extracted from the blockchain with data obtained from external sources can reveal one's identity. In his case study, Amin Aghaee [12] showed that it is possible to identify a specific user account within a network by utilizing information gathered from social media platforms. Law enforcement can use this approach to identify the cryptocurrency accounts of suspects.

(2) *Attacker Model 2—active access to network:* The third clustering approach presented by Teomete et al. (mentioned above) uses Internet Protocol addresses. Mining software needs to be injected into a network to listen to its traffic [88]. If two validators receive data from the same IP address, then it is assumed that this address belongs to the same user. Since it is possible to link IP addresses to real-life identities or locations, the user can be deanonymized [61]. Blockchain technology depends on the Internet infrastructure; otherwise, the connections between the different peers cannot be established. It is possible to interfere with the BGP to perform two types of *Routing Attacks*: (i) the *Partitioning Attack* and (ii) the *Delay Attack* [14, 31, 71, 96]. The Partitioning attack tries to split the Blockchain network into disjoint groups. This possibly enables applying double-spending attacks. The Delay attack creates false information to enforce the peers to resend their packages, leading to a delay in broadcasting blocks. Having a delay gives the attacker more time to find a block and thus receive the mining rewards.

(e) Blockchain Protocol. This section concerns all attacks specific to the blockchain network. We distinguish between five different attack models who have all the goal to exploit weaknesses to get coins. We use the verb *get* instead of *steal* in this context as the attackers in this scenario aim to get an advantage in the mining process and thus get more coins than other and not *steal* them from other users. The first attacker model requires that the TX is accepted by an user before it is validated or even if it is modified, meaning that the merchant accepts unconfirmed TXs. The second attacker model assumes that the attacker has more than 50% of the network's computing power. The third model is quite similar to the one before but generally requires the attacker to have enough computing power to perform a **Denial of Service (DoS)** attack. The fourth attacker model concerns networks that allow votes per IP address instead of per node. The attacker owns multiple IP address. In the fifth and last attacker model, the attacker needs a blackout or a downtime of the ISP.

(1) *Attacker Model 1—accepting unconfirmed TX:* One of the malicious schemes in the Blockchain is the so-called *double-spending*. Double spending refers to using the same coin multiple times [30, 31, 34, 41, 53, 54, 71, 73, 74, 85, 93, 96]. Double spending can appear when a chain is forked, and each of the forks spends the same coins but sends them to different addresses. Both transactions can be valid, but one of the forks will be abandoned as soon as one of the forks achieves a longer chain than the other. Usually, it is assumed that a double-spending attack can be carried out successfully if the attacker has a higher proportion of computing power than the honest party. Jang et al. [51] showed that even an attack with less than the assumed 50% can be profitable, which means that the profit made during the attack is greater than the costs to perform the attack. The study performed by Zhang et al. [97] showed that the success probability depends on the clustering coefficient and not on the path length or the number of nodes in the network. The lower the clustering-coefficient, the higher the success probability. Different attacks fall under the category of double-spending as the *Race Attack*, the *Finney Attack*, and the *Vector76 Attack*. The Race attack works when

a user accepts payments as soon as the TXs are sent but are not validated yet. An attacker can abuse it and create another TX, which transfers the coin to an address the attacker owns. The attacker publishes both TX in the network at the same time. This creates a race between the TXs, because both are valid, but the other becomes invalid if one gets mined in a block. If the attacker is lucky, then the TX to their address gets mined first, thus getting service without paying the merchant. The Finney attack is quite similar to the Race attack. The Finney attack has the same requirement as the Race attack; the merchant must accept the unconfirmed TX. The Finney attack requires the attacker to create two TXs and mine the malicious TX independently. The attacker releases the TX containing the transfer to the merchant to the network. The merchant checks the transaction and fulfills their part of the trade. The attacker releases the pre-mined block to the network as soon as the merchant accepts the payment, invalidating the merchant's transfer. The Vector76 attack extends the previously mentioned attacks by the possibility to apply this attack even when a merchant waits until a transaction is validated (by one block) and included in the Blockchain. The attacker mines a merchant-paying TX independently and withholds the TX to the network until they find a valid block. When the attacker finds a block, they publish the TX as soon as a new block is found by the network, thus creating a fork in the chain. The attacker publishes a malicious transaction in the fork, not including the merchant-paying TX, and double-spends the coin. The attack is successful when the fraudulent chain becomes longer than the chain with the merchant-paying TX [31, 41, 57, 71, 96].

Liu et al. [66] describe another attack based on the Blockchain protocol, the so-called *Malleability Attack*. This attack targets the way TX are created in the network. Since all network participants can receive broadcasted TX, they can also alter the TX and rebroadcast them. The transactions have to be valid, so the inputs and outputs cannot be changed, but it is possible to replace operators, e.g., `OP_0` with `OP_PUSHDATA2`, and add some operators to change its hash value. The altered and the original TX are competing now in the network, and in case the edited TX wins, the user can think that the TX failed. However, the TX was successful just with another hash value [66, 77].

Atzei et al. [16] published a survey on the attacks targeting the Ethereum network. The unpredictable state of the blockchain can lead to attacks as *GovernMental* and the exploitation of dynamic libraries. The problem is that it cannot be guaranteed that the contract will be in the same state when the transaction is performed as it was when the transaction was initiated.

(2) *Attacker Model 2—more than 50% of the network's computing power:* One of the most significant attacks is the *51% Attack* also called *Majority Attack* [71, 85, 93, 96]. This attack can be applied when the attackers have more than 50% of the computing power in the Blockchain network. By owning more computing power than the rest of the network, it is possible to change the existing Blockchain (to a certain degree, depending on the implementation of the Blockchain) by creating a fork where the attackers want to change the chain. This change can then lead to a double-spending attack or a reverse of validated transactions, or both.

(3) *Attacker Model 3—DoS attack:* It is possible to send vast amounts of data within the network to create DoS attacks [31, 96]. Those attacks can lead to a change in the rate of block creation, giving an advantage or a disadvantage to a group of miners. A DoS attack could also lead to double-spending.

(4) *Attacker Model 4—votes per IP address:* The *Sybil Attack* [31, 35, 71, 85, 96] is abusing the fact that some peer-to-peer networks allow a vote per IP address and not one vote per node. This allows an attacker to create multiple identities (IP addresses) to achieve a higher number of

votes within the network. When the attacker has more votes in the network, it gives the attacker the possibility to change the propagation in the network.

(5) *Attacker Model 5—blackout or downtime of ISP:* The *Eclipse Attack* [31, 47, 71, 85, 96] tries to populate all the ingoing and outgoing connections of a node by malicious nodes. This malicious table of IP addresses enables the attacker to influence the target perspective, since all information comes from malicious nodes. This attack needs a full restart of all its connections, which appears mostly after blackouts or when an ISP goes down.

(f) Others. This section summarizes all attacks that could not be assigned to any other category before, because, for example, they deal with a particular wallet type. All attackers have the goal to exploit implementation flaws to steal coins. The first attacker model has the additional goal to deanonymize users. In this model, the exchange platform itself is the attacker or is targeted. In the last case the exchange platform contains vulnerabilities that are exploitable by an attacker. These vulnerabilities are not considered in detail, as they are general vulnerabilities of web applications and servers. The second attacker model, deals generally with attacks focusing on implementation flaws, so the requisition is that there exist vulnerabilities on different levels that could be combined. The third attacker model concerns a specific kind of wallets, so-called air-gapped wallets. This model is divided into three different subcategories based on the technique used to gain access to the wallet. These are explained in more detail below.

(1) *Attacker Model 1—exchange platform:* Not only the security of the wallets but also the security of the exchange platforms is essential. Exchange platforms offer the exchange between money and cryptocurrencies. Since those platforms unite a massive amount of cryptocurrencies, they are targeted quite often. The exchange platform *Coinbase Exchange* has a volume of about $5.1 billion and is the biggest exchange provider in the U.S., offering fiat support for USD, EUR, and GBP [10].
Online wallets need a way to authenticate the user to the platform, which is often done via regular login credentials. If registration is needed to use a wallet, then the wallet provider can deanonymize the user [22], which means the platform could connect a wallet address to an explicit user.

(2) *Attacker Model 2—implementation flaws:* Arapinis et al. [15] state that each transaction includes three parties, the hardware (wallet), the client (software), and the user. Each component has its weaknesses. The wallet's security depends on the underlying cryptographic primitives, mainly the hash function and signature scheme. If one of them is broken, then it can result in a loss of coins. A broken hash function potentially results in a loss of receiving funds and a defective signature scheme in losing the spending funds. Moreover, the security heavily depends on the honesty of the client and user. Any input or output could be malicious if it compromised a client, and the security cannot be guaranteed anymore. In addition, all parties must adhere to their protocols and execute them correctly, but this could be thwarted by the user if, for example, they deviate from the intended security assumptions by using a too weak password. Furthermore, the security could be compromised by the missing usability of hash function comparison techniques.
Hierarchical deterministic wallets use a secret master-private key, or parent-private key, from which each child-private key can be generated pseudo-randomly. Accordingly, each child-public key can be generated from the master-public key or parent-public key. However, an attacker knowing the master-public key and any child-private key can recover the master-private key. Thus, the attacker could generate as many child-private keys as desired and spend the user's coins [44]. This attack is also known as *privilege escalation attack* [37].

(3) *Attacker Model 3—air-gapped wallets:* Davenport et al. [33] analyzed air-gapped wallet applications, which are known to be very secure due to their offline structure. Thus, it is difficult to compromise them. Davenport et al. identified two main challenges for attackers. The first one is the infiltration process that can be achieved in three different ways: (i) pre-downloaded infiltration, (ii) physical access, and (iii) remote infiltration. The infiltration consists of three steps. First, the online machine needs to be compromised, then some external media used for data transfer between the online machine and the air-gapped device, and last the air-gapped device itself. The second challenge is the exfiltration of data such as keys. This can be done by, e.g., using external media transfer or side-channel exfiltration. Since infiltration depends on an inside actor, it is harder to perform than exfiltration. Moreover, there exist more ways to exfiltrate than to infiltrate. However, both steps need to be successful for a successful attack. Guri [43] performed a similar analysis of key leakage in air-gapped wallet devices. He divided the infiltration into two categories, the post-installation infection, and the pre-installation infection. The former refers to a removable device such as a USB drive or an SD card to exchange information between offline (air-gapped) wallets and online wallets needed for a transaction. The attacker installed malicious software on the machine running the online wallet. When inserting the removable media, malware is installed on it. When the removable media is then inserted into the air-gapped device, it is also infected. The latter means installing a malicious or already infected wallet application. If the infiltration is successful, then the attacker establishes a covert channel to exfiltrate the keys. This can be accomplished using physical devices such as removable media, electromagnetic, electric, magnetic, optical, acoustic, or thermal data and signals.

After evaluating the different categories, we conclude that vulnerabilities and the related attacks assigned to the OS or software layer category pose the biggest threat to users. Attacking, for example, a mobile phone—and thus the wallet app—requires fewer resources than attacks assigned to one of the other categories and will succeed with a higher probability. In this context, resources are defined as computational and financial means as well as required technical background knowledge. Nevertheless, users can take some actions to better protect their wallets from attacks associated with the operating system or software layer, such as choosing a complex password to reduce the success of password cracking attacks. The user usually cannot exert any direct influence on the other levels. Blockchain attacks are more sophisticated in execution and require more resources. The same is arguable for network-level attacks. Since exploited vulnerabilities on these layers often mean that the entire system is affected and thus all users, not just some. Furthermore, it can lead to a total failure of the exchange provider. The amount of financial losses tends to be higher. They have a more severe impact, since one attack could lead to a complete loss of trust in the cryptocurrency and thus a currency's value is lost. Some of the attacks try to gain access to the funds secured by the wallet or get a financial benefit by delaying broadcasts to increase the chance to mine the block on their own. Others aim to obtain information about a specific user by deanonymizing the users by connecting a wallet to a person or IP address.

3.2 Countermeasures

In this section, we summarize the countermeasures obtained from the existing literature. This section is structured based on the main attacker goals, stealing coins and deanonymization, discussed in Section 3.1. The countermeasures are then further divided based on the different attacker models introduced in the section before. For example, the first attacker model of subsection (a) is referred to as a-1 and so on. Some fixes of the identified problems are already known and can be resolved quickly. Countermeasures do not exist against all attacker models. The last subsection (d) deals

with countermeasure that have not been covered in the literature from an attacker perspective as there is no attacker in this scenario. They tackle the problem of users forgetting or loosing their access credentials.

(a) Attacker Goal: Stealing Coins.

(1) *Unauthorized access, e.g., cracking passwords (a-1, a-3, b-1, b-2, c-1, d-1, f-1):* Brute-forcing passwords need a lot of computational power and time to be successful. This process can be even more challenging by implementing some of the following steps proposed by Khan et al. [57]. The most simple step is to use more complex passwords to increase the possibilities and thus the time needed to crack. Another attempt can be made to reduce the number of tries allowed by the user in a given time, and if exceeded, then the account can be locked, and the owner messages about the failed logins. A two-layer approach can be used to avoid this type of attack. The use of Captchas could avoid automatic login tries. In addition to that, they suggest using two or more login URLs and the modification of the ruleset of Hypertext access files. Increased awareness among users and developers could lead to the use of one or more techniques mentioned before.
Bulut et al. [28] provided a list of security objectives that developers and users should follow to secure the wallet applications. The wallet application must provide a sufficient level of authentication to prevent unauthorized access by offering two-factor authentication and ensuring that passwords are complex enough to be robust against brute-force and dictionary attacks. The communication channels and the storage of the wallets must be encrypted to avoid eavesdropping attacks and data leakage. Moreover, the receivers' addresses need to be displayed so that the user can visually verify them. Additionally, they should provide an integrity check mechanism and the option of a secure failure mode in case of a failure.
Since no legal foundation is responsible for cryptocurrencies, users are responsible for securing their virtual money. The users can do this by using sufficient authentication mechanisms. Therefore, Lim et al. [64] analyzed Bitcoin's security breaches to give an overview of countermeasures. Based on their findings they recommended applying at least two-factor authentication using Hardware Security Module or **One-Time Password (OTP)**.
The research conducted by Barber et al. [18] focused on vulnerabilities in the Bitcoin system and proposed solutions to the findings. The loss of Bitcoins can have several causes. It can result from malware attacks, which can be prevented by using standard threshold cryptography techniques. These techniques randomly split the private key and store the pieces in different locations. As another countermeasure, they introduced the principle of super wallets. These also use threshold cryptography techniques to divide the assets among multiple devices. The second reason for losing Bitcoins could be an accident or theft. They listed different solutions to this problem. The first is the simplest: using backups. If no backup is desired, then one can use pseudo-random keys. This means having a random secret master key from which private keys can be generated. To counteract wallet theft, encryption is a suitable solution. However, this countermeasure runs the risk of forgetting the decryption password, as it must be sufficiently complex.
Jasem et al. [52] introduced a new wallet consisting of a hot and cold wallet to countermeasure dictionary attacks and deanonymization. They modified the key generation algorithm by increasing the key space. Moreover, new keys are generated for each transaction, which decreases the risk of deanonymization.
The master seed is used to generate the private keys of a wallet. Jasim et al. [38] enhanced the security of the master seed of Brain wallets by adding entropy to it. They achieved this by including non-fixed ASCII encoded characters, which drastically decreases the success rate

of dictionary attacks as the increased entropy leads to the need of "building huge rainbow tables which are both space and time consuming process" [38].

Hu et al. [49] introduced a new approach to verifying the user's identity continuously in real-time using the user's mouse behavior biometrics in combination with a CNN network. Upon the first use of the wallet, the user's mouse behavior is monitored and passed on to the CNN model to generate a profile. In all subsequent logins the movements are measured and compared to the initial ones to verify the user's identity. The *BioWallet* [19] is another approach using biometric verification to increase the security of Bitcoin wallets. The wallet applies a two-step authentication process. In the initialization phase, the users have to scan their fingerprints and set a username and password. After that, the process starts with requesting the users for their fingerprint, which is then compared with the saved one for each transaction. If the fingerprints match, then the users are forwarded to the next step, entering their username and password. If those are correct, then the transaction will be performed. The users have three attempts to enter their fingerprint and password correctly. Otherwise, the process is aborted.

Zhu et al. [99] designed a new online wallet, called *HA-eWallet*. Usually, the private key of an online wallet is stored on the providers' server. If the server gets compromised, then all assets can be lost. The HA-eWallet prevents this by using three private keys out of five available keys to sign a transaction. Two encrypted keys are stored on the server in two different internal locations, and the other three keys are stored in the user node. So even if the server is compromised and an attacker obtains the keys stored on the server, it is not possible to perform a transaction.

(2) *No direct (physical) access to wallet (b-2, c-1, c-3):* The security of wallet applications can be improved by performing a more sophisticated security analysis before making it available to users. Takahashi et al. [87] developed a multi-layered approach consisting of a combination of three different methods. First, a static analysis is performed to find vulnerabilities in the app's source code. Second, the app is run in a sandbox environment to assess the runtime behavior, called dynamic analysis. The last method is a semantic analysis to evaluate the app's general behavior and decide whether it is a benign or malicious application.

It is possible to impersonate an RPC Server and, in some cases, even the client. Bui et al. [27] provided different ways to avoid these impersonations. One suggested solution is to avoid other users using the device containing the cryptocurrency wallet. Another improvement can be made by using RPC via TLS and by actively notifying the user when there is an issue starting the RPC server. Another approach is to change the architecture of the wallet from RPC to inter-process communication.

It is possible to prevent clickjacking by enabling Android's touch filter mechanism. However, this countermeasure is not applicable for scan-and-pay attacks presented by Ulqinaku et al. [91], since no click event is triggered. One possible way to prevent it would be to modify Android's OS system level by restricting views generated by a background service to a specific style to be easily recognizable for users. This measurement would require training the users to identify those styles and thus the malicious overlays. Another method would be to use sensitive views. The developers must implement these, who must mark the area where the QR code is displayed as sensitive. The operating system then automatically prevents any other application from creating an overlay at this point.

Dai et al. [32], Rezaeighaleh et al. [82], and Gentilal et al. [40] propose to divide the wallet applications into several individual sections, although their implementation of this idea differs. Dai et al. utilize the Trusted Execution Environment or, in their case, the **Secure Execution Environment (SEE)** based on the TrustZone scheme. The wallet is split into two layers, an

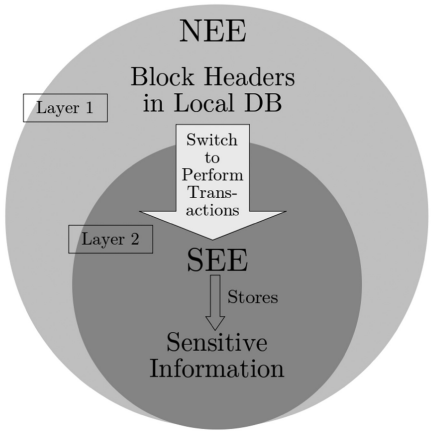


Fig. 4. Wallet model based on TrustZone scheme [32].

outer and an inner one, as shown in Figure 4. The inner one stores all sensitive information, such as the private and public keys, and performs the key generation process. This layer is the SEE and prevents the data from being stolen or tampered with. Only the block headers are stored in a local database located in the outer layer (Not secure Execution Environment). The operating system switches to safe mode for every transaction [32].

Gentilal et al.'s implementation is quite similar to the one mentioned above. It is also based on the TrustZone scheme. They are modifying an existing hardware wallet by moving its whole storage into the trusted zone. However, this causes a massive overhead for each writing or reading operation, since the whole storage needs to be decrypted before. Therefore, they deployed the write-cache, which uses intermediate storage. Gentilal et al. moved the most critical wallet functions into the secure world to fully protect the private key from being stolen. These include all operations dealing with sensitive data, the key generation, and the signing process. Moreover, the random number generator is also used in the secure world. Due to these modifications, attackers can no longer manipulate sensitive data and the wallet's code [40].

Rezaeighaleh et al. developed a three-layered model, illustrated in Figure 5, to enhance the security of wallet applications. The first layer, the offline layer, functions as a backup wallet for the second layer, containing the protected or superior wallet. This is the central component, since it is responsible for generating and storing the master seed and the keys. If the user wants to spend some coins, then the wallet automatically switches to the third layer, the online layer. It receives a subordinated seed from the superior wallet and only holds a limited fund so that in the event of an attack, only this limited amount of coins gets lost. Two additional security layers are implemented by splitting the wallet into those three levels, since the backups are offline and cannot be accessed. The online component is not holding the master seed and the keys and only a limited amount of coins.

Furthermore, Rezaeighaleh et al. [81] recommended creating more than one backup wallet. They designed a new wallet application consisting of two wallets, the *main wallet* and the *backup wallet*. The backup wallet generates a key pair (pk, sk), computes the public key's verification code, and exports the pk. After that, the main wallet receives pk and calculates the verification code to compare both verification codes. If matching, then the main wallet generates the key pair and computes the transport key to encrypt the master seed. The

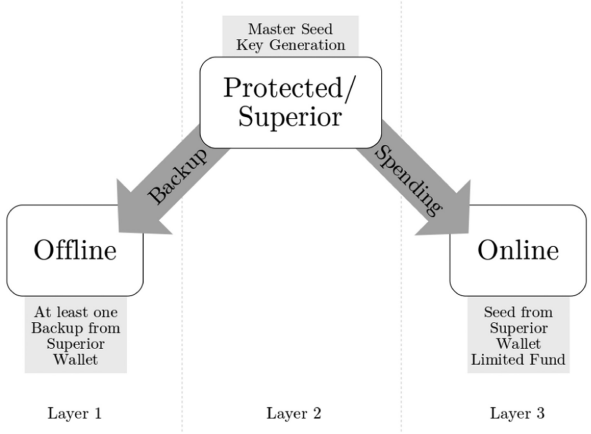


Fig. 5. Three-layered wallet scheme [82].

backup wallet imports the encrypted master seed and pk. It computes the transport key itself and thus can decrypt the master seed.

Homoliak et al. [48] proposed a similar approach for Ethereum-based wallets. They combined three components to create a safer and more user-friendly system to prevent ether loss due to attacks. The first component is an air-gapped authenticator that is used to verify the transaction using OTPs for each one. The second component is the client, which is used for user interaction and to trigger a transaction. The last one is the smart contract, which, when verified, executes the transactions.

Hu et al. [50] developed an additional mobile application (Bitcoin Security Rectifier) addressing the vulnerabilities introduced by the Bitcoin client library *BitcoinJ*. The app fixes the weaknesses without changing Bitcoin's protocol or the wallet applications themselves. The Bitcoin Security Rectifier examines all ingoing and outgoing Bitcoin messages and applies additional filtering rules to prevent arbitrary transactions.

Gutoski et al. [44] introduced a new HD wallet countering the recovery problem of the master-private key for known master-public key and child-private key. They propose to use m master-private keys for the generation of child-private keys. Therefore, to recover the master-private keys, the attacker requires m child-private keys, which makes breaking the wallet "at least as hard as the so-called 'one more' discrete logarithm problem." Fan et al. [37] focused on solving the same vulnerability. They proposed to use the trapdoor hash of a child-private key for singing instead of the child-private key itself. Even if the parent-public key and any trapdoor hashed child-private key is known to an attacker, it is not possible to recover the parent-private key.

(b) Attacker Goal: Getting Coins.

(1) *Accepting unconfirmed TX (e-1):* Another way of securing the use of wallet applications was introduced by Bamert et al. [17]. The so-called BlueWallet is a hardware token for authorization of Bitcoin transactions. It can be used to combine the user's smartphone or computer running a wallet application to sign the transactions securely. The BlueWallet is connected to the device running the wallet via Bluetooth Low Energy. The unsigned TX is transmitted to the BlueWallet and verified there. If the TX was accepted, then the user must enter a preset password to trigger the signing action. The signed TX is then transmitted back to the device.

(2) *Malleability Attack (e-1):* Rajput et al. [78] invented a modification to the Bitcoin system to prevent Malleability attacks. They appended the hash of the intermediate raw transaction to the transaction id, resulting in $hash_{prev} \mid hash_{new}$. If an attacker now changes $hash_{prev}$ to perform a Malleability attack, as the one known from Mt.Gox, then the transaction will be performed normally but the sender can know verify that the transaction arrived at the receiver searching for the appended $hash_{new}$ of the transaction id. If the attacker modifies the $hash_{new}$ part, then the transaction will be discarded by the miners.

(c) Attacker Goal: Deanonymizing Users. We combined the active and passive access as the countermeasures preventing active access can also prevent the successful use of passive access.

(1) *Passive and active access to network (d-1, d-2):* Venkatakrishnan et al. [24] created a new networking policy, called *Dandelion*, to prevent deanonymization in the Bitcoin network by redesigning it. The network is protected against adversaries using botnets "by mixing messages from different users on a graph that is unknown to the adversary." Thus, it is nearly impossible for an adversary to reveal an user's identity.
Bergman et al. [20] presented three wallet applications that focus on anonymization. The first one is the Darkwallet [2], which uses a stealthy address instead of the usually used hashed address. This means that a new public key is generated for each transaction based on a fixed, known public key. The second wallet is the Wasabi Wallet [11], which utilizes the Chaumian CJ transactions, which uses blind signatures. This means a third-party tumbler validates the transactions so that never the receiver and the sender are known at the same time. The last wallet presented is the Samourai Wallet [9]. It generates a new address for each outside transaction and sends the payments through multiple dummy addresses to conceal the user's address.
Rai et al. [76] introduced a new wallet application primarily designed for Linux systems, which uses different receiving addresses for each transaction to improve the users' privacy. The seed is represented as a 12 words long mnemonic passphrase. The wallet is open source.

(d) Key Recovery Approaches. If users lose or forget their access credentials to their wallets, then they can no longer access and consequently use their coins. This can be avoided by using key recovery methods. Rakdej et al. [79] presented an approach extending an existing wallet with a retire option to transfer the **Unspent Transaction Output (UTxO)** to a backup address and thus preventing the loss of all assets. The UTxO is the amount of coins left after a performed transaction and used as input for subsequent transactions. The retire option, if enabled, sets a maximum time range that is allowed between two logins. If this time range is exceeded, then the UTxO will be transferred to the predefined backup address and accessed from there. The backup address could belong to the wallet of a friend, partner, or family member. The countdown of the time range is reset after each login.

Soltani et al. [84] introduced a new protocol for secure private key recovery. The architecture of the proposed method is based on the Hyperledger Indy in combination with a digital wallet and Key Escrow Providers. *Hyperledger Indy* is a distributed ledger software enabling "[interoperability] across administrative domains, applications, and any other silo" [5]. In the first step, the wallet generates multiple secret keys. Based on these, the private key can later be recovered. The secret keys are signed using different temporary keys. Data bundles are then sent to the Key Escrow Providers, each containing a signed secret key. The wallet establishes a secure communication channel to the Key Escrow Providers and receives the data bundles containing the signed secret keys to recover the private key. After verifying the signature of each of these keys, the secret keys are used to reconstruct the required private key.

Liu et al. [65] designed a new key management system for Bitcoin wallets with the option of easy key recovery. The private key generation consists of a combination of a random seed and an easy-to-remember passphrase. Only the list of random seeds needs to be stored locally. The private keys are then generated using the passphrase. Without this, it is not possible to generate any private key. If the passphrase is forgotten, then it is possible to recover the keys by answering preset personal questions.

In summary, most of the analyzed problems can be avoided and countered but still need more attention among the stakeholders. Nevertheless, there are still some open issues based on the network layer, since changes there would also affect other traffic on the Internet.

The end-user can apply some countermeasure on his own. Splitting up cryptocurrency into multiple wallets can avoid the complete loss of assets. A combination of hot and cold wallets should be used. Moreover, it is even better to use the proposed multi-layer architecture, containing a master wallet, an offline backup, and a wallet for online spending. The user should be aware that a confirmed transaction can still be altered due to a change in the leading chain and should wait for several confirmations to decrease the likelihood of the change. The user should be aware of the possibilities of deanonymization. Using different entry nodes when broadcasting TX can help to avoid deanonymization based on the broadcasting delays.

Developers should be aware of widespread problems, i.e., storing and generating the key pairs encrypted and not in plaintext and not using hard-coded encryption keys. Zhang et al. [98] proposed an extended signing protocol that is focusing on safe key storage on devices. Most secret sharing protocols are not designed to mitigate key reconstruction attacks. Therefore, their protocol focuses on this problem. In addition to that, we advise using the SEE to avoid access through other applications and block accessibility tools during critical processes (key generation, transfers). Privacy can be increased in some wallet applications by not having separate login details, so the platform owner does not know the actual owner of the wallet. However, this can be necessary due to legal restrictions. Wallet applications can integrate techniques as trickling or diffusion to boost the privacy of the users.

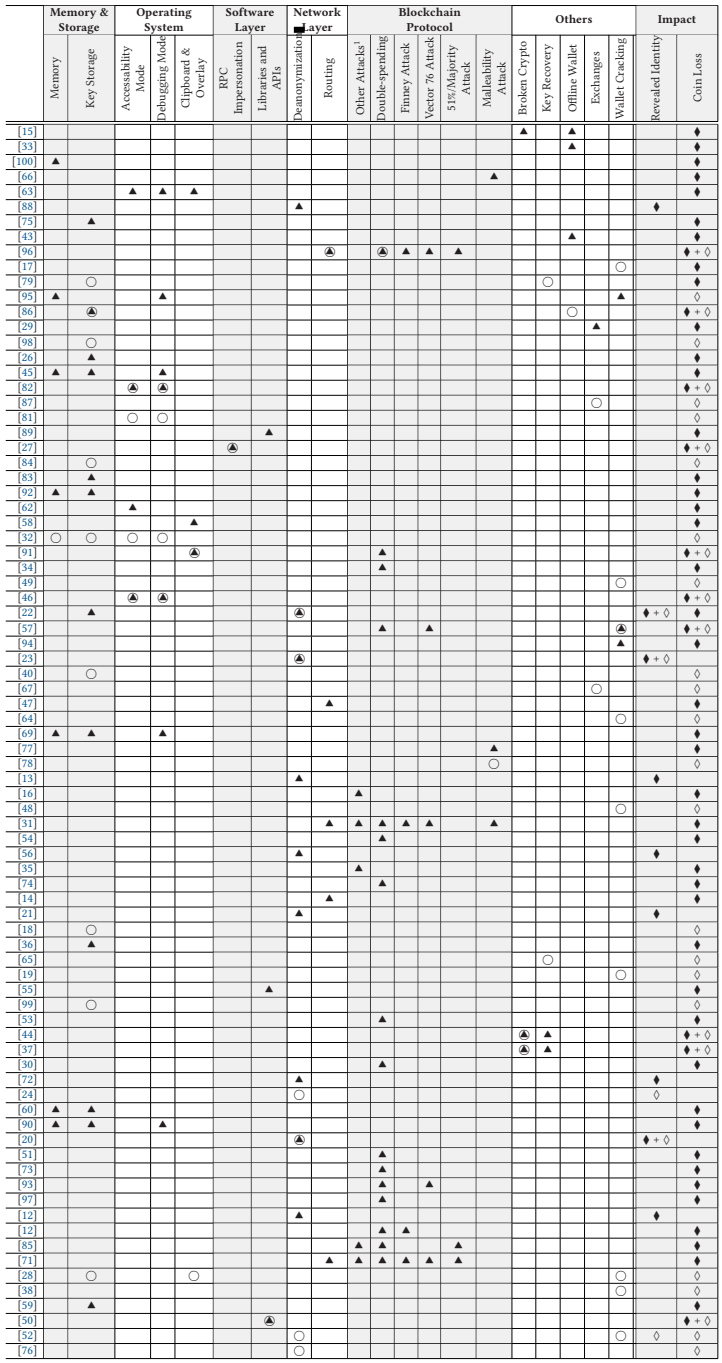
Table 2 visualizes the topics of the selected publications based on the attack categories denoting whether only attacks (▲), only countermeasures(◯), or both (◮) are covered. Moreover, the last two columns indicate the described attacks' (negative = ⧫) and countermeasures' (positive = ◊) impact.

4 FUTURE RESEARCH

In this section, we propose future research directions based on our analysis of the existing literature.

(1) The discrepancy between potential protection measures and their actual application in available wallet applications is significant. Based on the publications we evaluated, it is not clear why some of the countermeasures are not applied. To gain an insight into why developers do not apply those methods, it would be necessary to analyze how complicated it would be to apply some of the proposed defenses to existing applications and, if necessary, adjust them based on the results of this analysis.
(2) Providing a framework containing vulnerabilities and a description of the possible defenses would make it easier for the developers to implement the protection mechanisms. This could be combined with a tool to automatically assess the security level of an application by identifying the presence of known weaknesses and thus making it easier for the developers to fix the vulnerabilities.

Table 2. Attacks and Countermeasures per Publication



▲ = only Attacks, ◯ = only Countermeasures, ⏏ = Attacks & Countermeasures.
♦ = only negative Impact, ◊ = only positive Impact, ♦ + ◊ = negative & positive Impact.
[1] = *Sybil Attack, Eclipse Attack.*

(3) Some attacks explained in parts Section 3.1 (a) and Section 3.1 (b), such as the incorrect storage of keys, insecure passwords, third-party keyboards, and clipboard-related vulnerabilities, could be mitigated by providing adequate information to the user to raise awareness and offering a solution they can apply themselves. For example, by providing a definition of strong passwords or notifications about the risks of using a third-party keyboard, including a description on how to change that.
(4) As mentioned earlier, most of the literature on mobile wallets focuses on Android-based applications, so there is a gap in research about vulnerabilities specific to iOS. Therefore, more research is needed on wallet applications for iOS devices.

5 CONCLUSION

Cryptocurrencies are gaining prominence within the private sector, including companies and individuals. Both interest groups use cryptocurrency applications. Hence, both groups need to have an overview of the existing attacks and countermeasures to protect themselves. Moreover, developers need to be up-to-date regarding the possible attacks and, even more importantly, the available defenses. Therefore, we provide a structured review of identified cryptocurrency wallets' vulnerabilities and a taxonomy of attacks and defenses.

Our research concludes that wallet attacks targeting the Operating System and the Software layer are the most common ones from the user's perspective. Regarding those attacks, the user can improve the security directly by choosing a suitable wallet application that includes using more than one wallet application, i.e., combining an offline and online wallet.

Our review showed that even if countermeasures against specific attacks exist, they are often not adopted in the available wallet applications. It is necessary to investigate why this is the case to improve the security. On top of this, providing a framework of known vulnerabilities and their countermeasures, in conjunction with an automated security assessment tool, could further improve the adoption of protection mechanisms and thus the security of wallet applications.

The following two examples can illustrate the problem mentioned above of adopting defenses. First, anonymity is one of the reasons why people use cryptocurrencies like Bitcoin. However, there are some known techniques to reveal the identity of a user. Nevertheless, there exist also practical ways to make these attacks more difficult, for example, by using a new address for each new transaction. Our review of the literature has shown that only a few of the available applications use such methods. Second, users of security-critic applications such as crypto wallets would usually assume the developers of those applications to have an in-depth knowledge of security and cryptography. However, many applications store their keys in plain, leading to high-security risks to users if an attacker gains access to the storage and thus can steal all coins [45, 69, 90]. Some keys were also stored in a way that no root access was needed to access them [100]. Even if this threat can easily be mitigated by using encryption, many wallet applications developers did not apply any techniques.

Since this research area is still relatively new with a growing interest from individuals, companies, law enforcement, and criminals, it offers excellent potential for further research. Further research can include additional sources such as white papers, audits, and online forums, since they provide many resources. However, they can firmly be biased to boost or defame a currency or cryptocurrency application, so each source's quality has to be evaluated more critically. Moreover, there are hardly any publications concerning cryptocurrency web applications, which implies that further research is needed. The same lack of literature is evident in mobile wallets when considering apps based on iOS, as most of the work concerns Android apps.

APPENDIX

A METHOD

The complete process of information gathering and extracting is described in this section. The inclusion and exclusion criteria are clearly defined.

A.1 Data Sources

This systematic literature review sources include the databases Scopus, ACM, and IEEE, since those offer many publications in computer science and cybersecurity. We used the database Google Scholar for the backward and forward reference checking. We excluded white papers, yellow papers, audits, and blogs, since they are often biased toward a specific cryptocurrency or cryptocurrency wallet application. Our study should give an unbiased perspective of vulnerabilities and countermeasures in general and specific wallet applications. Vulnerabilities found in a particular wallet application may be relevant to the security of other applications.

A.2 Study Selection Process

The study selection process is divided into two parts. The first part of the process is visualized in Figure 6 and includes applying search queries to the three databases Scopus, ACM, and IEEE. The second part consists of backward and forward checking of the references of the publications found in the previous part. We explain the two parts in more detail below.

Part 1: Database Search Queries. The first step was to define the search queries in Table 3 and then apply them to the databases Scopus, ACM, and IEEE. The formation of the final search queries is explained below in paragraph A.2. Next, we combined all the results of the different databases into one, which resulted in 720 studies. The next step was the deletion of duplicates. After the duplicate removal, the number of studies was reduced to 567 publications. The 557 publications were filtered after their release date. We decided to set the boundary to the year 2009, since this was the year in which Bitcoin, the first cryptocurrency, went online. Every publication before the year 2009 was not considered. The resulting 534 studies were evaluated whether their abstract fit our study and filtered out if not based on the A.2, leading to 61 publications. The next step was to read the conclusion of the publications to get a better understanding of the publications without reading all of them thoroughly. After this screening, 42 publications were left to read the full study. After the complete reading, we ended up with 38 relevant publications.

Keywords. The first part of the study selection process used a two-step approach to collect an initial overview of this field's existing research. The first step was to use a very detailed search query, which led to many results but insufficient references to the research questions. So we decided to extend the search by splitting the request into two additional search queries addressing the two different perspectives (Mobile Security and Cryptocurrency Wallets). The complete queries can be found in Table 3.

Part 2: Backward-Forward Reference Checking. We performed the backward and forward checking for the 38 resulting papers of A.2, starting with the backward checking. This means searching the references of all selected publications for additional matching ones based on the A.2 below. After that, we performed the forward checking of the 38 publications, which means checking the papers that cited them, and also applied the A.2. We did not apply any filtering regarding the publication before 2009.

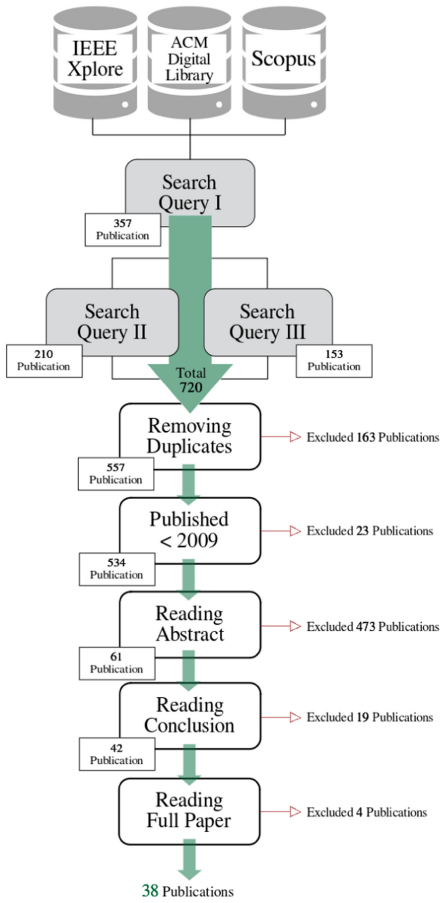


Fig. 6. Database search queries process.

Table 3. Search Queries

Query	AND			
1.	Crypto OR Cryptocurrenc* OR Blockchain OR Bitcoin OR Ethereum	Wallet	Android OR iOS OR Smartphone	APP OR Application
2.	Application	Wallet	Security	
3.	Cryptocurrency	Wallet		

Inclusion and Exclusion Criteria. We clearly defined our exclusion and inclusion criteria to achieve a reproducible, comprehensible, and systematic literature review. A study is included if it is related to at least one of the following topics:

(1) Cryptocurrency in general
(2) Cryptocurrency wallets
(3) Application security, including

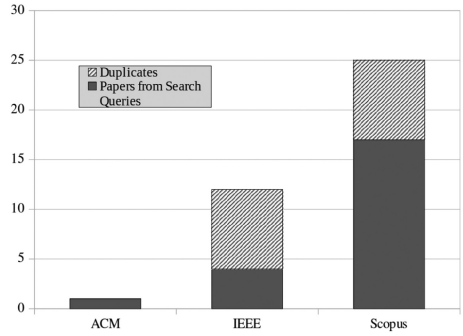


Fig. 7. Included publications per source.

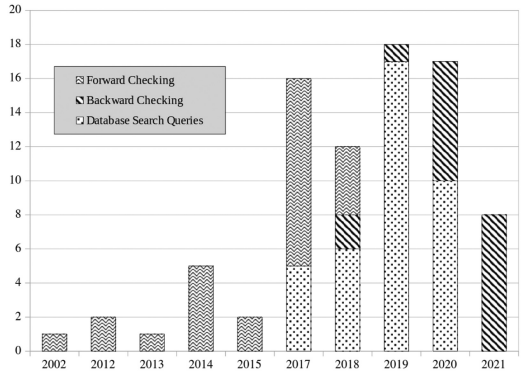


Fig. 8. Publications over time.

(a) Attacks on mobile applications
(b) Attacks on cryptocurrencies
(c) Security breaches on wallets
(d) Security problems in respect to the Blockchain technology

Studies not related to any of the above topics are excluded.

A.3 Selected Publications

Figure 7 gives an overview of the distribution of the chosen publications concerning the databases used. ACM returned only two relevant publications, while IEEE provided 15 and Scopus 33 publications. However, there were 11 overlapping publications in the two last databases and one overlapping publication between ACM and Scopus, resulting in 38 publications.
As shown in Figure 8, most of the selected publications are in a time frame between 2017 and 2021, highlighting the issue's topicality. Which in turn ensures the up-to-dateness of the information collected. The search queries produced 38 papers (dotted), the backward reference checking 27 (hashed), and the forward reference checking 17 (waved) publications, which results in 82 publications in total.

ACKNOWLEDGMENTS

We express our gratitude to our supervisor during the process of our Master's thesis, Sundas Munir, at Halmstad University. This work was partially supported by the Wallenberg AI,

Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

REFERENCES

[1] Blockchain Explorer—Search the Blockchain. Retrieved October 29, 2021 from https://www.blockchain.com/charts/my-wallet-n-users.

[2] GitHub. Darkwallet/darkwallet: Your Keys, Your Privacy, Your Sovereignty. Retrieved November 30, 2021 from https://github.com/darkwallet/darkwallet.

[3] hashcat. Advanced Password Recovery. Retrieved November 30, 2021 from https://hashcat.net/hashcat/.

[4] Huobi Wallet. Professional Multi-currency Wallet. Retrieved November 30, 2021 from https://www.huobiwallet.com/en/.

[5] Hyperledger Indy–Hyperledger Foundation. Retrieved November 29, 2021 from https://www.hyperledger.org/use/hyperledger-indy.

[6] imToken. Ethereum & Bitcoin Wallet. Retrieved November 30, 2021 from https://token.im/.

[7] Statista. Mobile OS Market Share 2021. Retrieved November 10, 2021 from https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/.

[8] GitHub. Rockyou.txt wordlist. Retrieved November 30, 2021 from https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt.

[9] Samourai Wallet. Retrieved November 30, 2021 from https://samouraiwallet.com/.

[10] CoinMarketCap. Top Cryptocurrency Exchanges Ranked by Volume. Retrieved November 29, 2021 from https://coinmarketcap.com/rankings/exchanges/.

[11] Wasabi Wallet. Bitcoin Privacy Wallet with Built-in CoinJoin. Retrieved November 30, 2021 from https://wasabiwallet.io/.

[12] Amin Aghaee. 2021. Identifying blockchain-based cryptocurrency accounts using investment portfolios. arXiv: 2110.04394. Retrieved from https://arxiv.org/abs/2110.04394.

[13] Elli Androulaki, Ghassan O. Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. 2013. Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*. Springer, 34–51.

[14] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. 2017. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'17)*. IEEE, 375–392.

[15] M. Arapinis, A. Gkaniatsou, D. Karakostas, and A. Kiayias. 2019. A formal treatment of hardware wallets. 426–445. https://doi.org/10.1007/978-3-030-32101-7_26 Conference Paper.

[16] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. 2017. A survey of attacks on ethereum smart contracts (sok). In *International Conference on Principles of Security and Trust*. Springer, 164–186.

[17] T. Bamert, C. Decker, R. Wattenhofer, and S. Welten. 2014. BlueWallet: The secure bitcoin wallet. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 8743, 65–80.

[18] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. 2012. Bitter to better-how to make bitcoin a better currency. In *International Conference on Financial Cryptography and Data Security*. Springer, 399–414.

[19] E. Benli, I. Engin, C. Giousouf, M. A. Ulak, and S. Bahtiyar. 2017. BioWallet: A biometric digital wallet. In *Proceedings of the 12th International Conference on Systems*. 23–27.

[20] Karolina Bergman and Saeed Rajput. 2021. Revealing and concealing bitcoin identities: A survey of techniques. In *Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure*. 13–24.

[21] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. 2014. Deanonymisation of clients in bitcoin P2P network. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. 15–29.

[22] A. Biryukov and S. Tikhomirov. 2019. Security and privacy of mobile wallet users in bitcoin, dash, monero, and zcash. *Perv. Mobile Comput.* 59 (2019). https://doi.org/10.1016/j.pmcj.2019.101030

[23] A. Biryukov and S. Tikhomirov. 2019. Transaction clustering using network traffic analysis for bitcoin and derived blockchains. In *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'19)* (2019), 204–209. https://doi.org/10.1109/INFCOMW.2019.8845213

[24] Shaileshh Bojja Venkatakrishnan, Giulia Fanti, and Pramod Viswanath. 2017. Dandelion: Redesigning the bitcoin network for anonymity. *Proc. ACM Meas. Anal. Comput. Syst.* 1, 1 (2017), 1–34.

[25] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W Felten. 2015. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 104–121.

[26] M. Brengel and C. Rossow. 2018. Identifying key leakage of bitcoin users. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 11050, 623–643. https://doi.org/10.1007/978-3-030-00470-5_29

[27] T. Bui, S.P. Rao, M. Antikainen, and T. Aura. 2019. Pitfalls of open architecture: How friends can exploit your cryptocurrency wallet. In *Proceedings of the 12th European Workshop on Systems Security (EuroSec'19)*. https://doi.org/10.1145/3301417.3312495

[28] Yasir Emre Bulut and İsa Sertkaya. 2020. Security problem definition and security objectives of cryptocurrency wallets in common criteria. *Biliş. Teknol. Derg.* 13, 2 (2020), 157–165.

[29] J. J. Castonguay and S. Stein Smith. 2020. Digital assets and blockchain: Hackable, fraudulent, or just misunderstood?*. *Account. Perspect.* 19, 4 (2020), 363–387. https://doi.org/10.1111/1911-3838.12242

[30] Usman W. Chohan. 2021. The double spending problem and cryptocurrencies. Available at SSRN 3090174.

[31] Mauro Conti, E. Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. 2018. A survey on security and privacy issues of bitcoin. *IEEE Commun. Surv. Tutor.* 20, 4 (2018), 3416–3452.

[32] W. Dai, J. Deng, Q. Wang, C. Cui, D. Zou, and H. Jin. 2018. SBLWT: A secure blockchain lightweight wallet based on trustzone. *IEEE Access* 6 (2018), 40638–40648. https://doi.org/10.1109/ACCESS.2018.2856864

[33] A. Davenport and S. Shetty. 2019. Air gapped wallet schemes and private key leakage in permissioned blockchain platforms. In *Proceedings of the 2nd IEEE International Conference on Blockchain (Blockchain'19)*, 541–545. https://doi.org/10.1109/Blockchain.2019.00004

[34] A. Dmitrienko, D. Noack, and M. Yung. 2017. Secure wallet-assisted offline bitcoin payments with double-spender revocation. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIA CCS'17)*, 520–531. https://doi.org/10.1145/3052973.3052980

[35] John R. Douceur. 2002. The sybil attack. In *International Workshop on Peer-to-peer Systems*. Springer, 251–260.

[36] Shayan Eskandari, David Barrera, Elizabeth Stobert, and Jeremy Clark. 2015. *A First Look at the Usability of Bitcoin Key Management*. DOI: 10.14722/used.2015.23015

[37] Chun-I Fan, Yi-Fan Tseng, Hui-Po Su, Ruei-Hau Hsu, and Hiroaki Kikuchi. 2020. Secure hierarchical bitcoin wallet scheme against privilege escalation attacks. *Int. J. Inf. Secur.* 19, 3 (2020), 245–255.

[38] F. Jasem. 2019. *Enhancing the Security of the Bitcoin Wallet Master Seed*. DOI: 10.4206/aus.2019.n26.2.71

[39] Amir Feder, Neil Gandal, J. T. Hamrick, and Tyler Moore. 2017. The impact of DDoS and other security shocks on bitcoin currency exchanges: Evidence from Mt. Gox. *J. Cybersecur.* 3, 2 (2017), 137–144.

[40] Miraje Gentilal, Paulo Martins, and Leonel Sousa. 2017. TrustZone-backed bitcoin wallet. In *Proceedings of the 4th Workshop on Cryptography and Security in Computing Systems (CS2'17)*. Association for Computing Machinery, New York, NY, 25–28. https://doi.org/10.1145/3031836.3031841

[41] Arunima Ghosh, Shashank Gupta, Amit Dua, and Neeraj Kumar. 2020. Security of cryptocurrencies in blockchain technology: State-of-art, challenges and future prospects. *J. Netw. Comput. Appl.* 163 (2020), 102635.

[42] Steven Goldfeder, Harry Kalodner, Dillon Reisman, and Arvind Narayanan. 2018. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. *Proceedings on Privacy Enhancing Technologies*, Vol. 4, 179–199.

[43] M. Guri. 2018. BeatCoin: Leaking private keys from air-gapped cryptocurrency wallets. In *Proceedings of the IEEE International Conference on Internet of Things (iThings'18); IEEE Green Computing and Communications (GreenCom'18); IEEE Cyber, Physical and Social Computing (CPSCom'18), and IEEE Smart Data (SmartData'18)*. 1308–1316. https://doi.org/10.1109/Cybermatics_2018.2018.00227

[44] Gus Gutoski and Douglas Stebila. 2015. Hierarchical deterministic bitcoin wallets that tolerate key leakage. In *International Conference on Financial Cryptography and Data Security*. Springer, 497–504.

[45] Trevor Haigh, Frank Breitinger, and Ibrahim Baggili. 2018. If i had a million cryptos: Cryptowallet application analysis and a trojan proof-of-concept. In *International Conference on Digital Forensics and Cyber Crime*. Springer, 45–65.

[46] D. He, S. Li, C. Li, S. Zhu, S. Chan, W. Min, and N. Guizani. 2020. Security analysis of cryptocurrency wallets in android-based applications. *IEEE Netw.* 34, 6 (2020), 114–119. https://doi.org/10.1109/MNET.011.2000025

[47] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. 2015. Eclipse attacks on bitcoin's peer-to-peer network. In *Proceedings of the 24th USENIX Security Symposium (USENIX Security'15)*. 129–144.

[48] I. Homoliak, D. Breitenbacher, O. Hujnak, P. Hartel, A. Binder, and, P. Szalachowski. 2020. SmartOTPs: An air-gapped 2-factor authentication for smart-contract wallets. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 145–162.

[49] Teng Hu, Xiaolei Liu, Weina Niu, Kangyi Ding, Yanping Wang, and Xiaosong Zhang. 2020. Securing the private key in your blockchain wallet: A continuous authentication approach based on behavioral biometric. In *Journal of Physics: Conference Series*, Vol. 1631. IOP Publishing, 012104.

[50] Yiwen Hu, Sihan Wang, Guan-Hua Tu, Li Xiao, Tian Xie, Xinyu Lei, and Chi-Yu Li. 2021. 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*. 89–100.

[51] Jehyuk Jang and Heung-No Lee. 2020. Profitable double-spending attacks. *Appl. Sci.* 10, 23 (2020), 8477.

[52] Farah Maath Jasem, Ali Makki Sagheer, and Abdullah M. Awad. 2021. Enhancement of digital signature algorithm in bitcoin wallet. *Bull. Electr. Eng. Inf.* 10, 1 (2021), 449–457.
[53] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. 2012. Double-spending fast payments in bitcoin. In *Proceedings of the ACM Conference on Computer and Communications Security*. 906–917.
[54] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. 2012. Two bitcoins at the price of one? Double-spending attacks on fast payments in bitcoin. *Cryptol. EPrint Arch.* (2012).
[55] Puneet Kumar Kaushal, Amandeep Bagga, and Rajeev Sobti. 2017. Evolution of bitcoin and security risk in bitcoin wallets. In *Proceedings of the International Conference on Computer, Communications and Electronics (Comptelix'17)*. IEEE, 172–177.
[56] Merve Can Kus Khalilov and Albert Levi. 2018. A survey on anonymity and privacy in bitcoin-like digital cash systems. *IEEE Commun. Surv. Tutor.* 20, 3 (2018), 2543–2585.
[57] A. G. Khan, A. H. Zahid, M. Hussain, and U. Riaz. 2019. Security of cryptocurrency using hardware wallet and QR code. In *Proceedings of the 3rd International Conference on Innovative Computing (ICIC'19)*. https://doi.org/10.1109/ICIC48496.2019.8966739
[58] C. Y. Kim and K. Lee. 2018. Risk management to cryptocurrency exchange and investors: Guidelines to prevent potential threats. In *Proceedings of the International Conference on Platform Technology and Service (PlatCon'18)*. https://doi.org/10.1109/PlatCon.2018.8472760
[59] Ju-Seong Ko and Jin Kwak. 2020. Private key recovery on bitcoin with duplicated signatures. *KSII Trans. Internet Inf. Syst.* 14, 3 (2020), 1280–1300.
[60] Wiebe Koerhuis, Tahar Kechadi, and Nhien-An Le-Khac. 2020. Forensic analysis of privacy-oriented cryptocurrencies. *Forens. Sci. Int.: Digit. Invest.* 33 (2020), 200891.
[61] Philip Koshy, Diana Koshy, and Patrick McDaniel. 2014. An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security*. Springer, 469–485.
[62] Y. Leguesse, M. Vella, C. Colombo, and J. Hernandez-Castro. 2020. Reducing the forensic footprint with android accessibility attacks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 12386, 22–38. https://doi.org/10.1007/978-3-030-59817-4_2
[63] C. Li, D. He, S. Li, S. Zhu, S. Chan, and Y. Cheng. 2020. Android-based cryptocurrency wallets: Attacks and countermeasures. In *Proceedings of the IEEE International Conference on Blockchain (Blockchain'20)*, 9–16. https://doi.org/10.1109/Blockchain50366.2020.00010
[64] Il-Kwon Lim, Young-Hyuk Kim, Jae-Gwang Lee, Jae-Pil Lee, Hyun Nam-Gung, and Jae-Kwang Lee. 2014. The analysis and countermeasures on security breach of bitcoin. In *International Conference on Computational Science and Its Applications*. Springer, 720–732.
[65] Yi Liu, Ruilin Li, Xingtong Liu, Jian Wang, Lei Zhang, Chaojing Tang, and Hongyan Kang. 2017. An efficient method to enhance bitcoin wallet security. In *Proceedings of the 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID'17)*. IEEE, 26–29.
[66] Y. Liu, X. Liu, L. Zhang, C. Tang, and H. Kang. 2017. An efficient strategy to eliminate malleability of bitcoin transaction. In *Proceedings of the 4th International Conference on Systems and Informatics (ICSAI'17)*, 960–964. https://doi.org/10.1109/ICSAI.2017.8248424
[67] P. McCorry, M. Möser, and S. T. Ali. 2018. *Why Preventing a Cryptocurrency Exchange Heist Isn't Good Enough*, 225–233. https://doi.org/10.1007/978-3-030-03251-7_27
[68] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2013. A fistful of bitcoins: Characterizing payments among men with no names. In *Proceedings of the Conference on Internet Measurement Conference*. 127–140.
[69] Angelica Montanez. 2014. Investigation of cryptocurrency wallets on iOS and android mobile devices for potential forensic artifacts. Department of Forensic Science, Marshall University, Huntington, WV.
[70] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. *Decentr. Bus. Rev.* (2008), 21260.
[71] T. M. Navamani. 2021. A review on cryptocurrencies security. *J. Appl. Secur. Res.* (2021), 1–21.
[72] Till Neudecker and Hannes Hartenstein. 2017. Could network information facilitate address clustering in bitcoin?. In *International Conference on Financial Cryptography and Data Security*. Springer, 155–169.
[73] Kervins Nicolas, Yi Wang, George C. Giakos, Bingyang Wei, and Hongda Shen. 2020. Blockchain system defensive overview for double-spend and selfish mining attacks: A systematic approach. *IEEE Access* 9 (2020), 3838–3857.
[74] A. Pinar Ozisik and Brian Neil Levine. 2017. An explanation of nakamoto's analysis of double-spend attacks. arXiv:1701.03977. [Online]. Available: http://arXiv.org/abs/1701.03977.
[75] P. Praitheeshan, Y. W. Xin, L. Pan, and R. Doss. 2019. Attainable hacks on keystore files in ethereum wallets—A systematic analysis. *Commun. Comput. Inf. Sci.* 1113 (2019), 99–117.
[76] Deepti Rai, Manjesh P. Shetty, Gautham L. Alva, Akshith Hegde, and Mohammed Shiran. 2018. Wallet for bitcoin cryptocurrency. *Int. Res. J. Eng. Technol.* 05, 07 (2018).

[77] Ubaidullah Rajput, Fizza Abbas, Rasheed Hussain, Hasoo Eun, and Heekuck Oh. 2014. A simple yet efficient approach to combat transaction malleability in bitcoin. In *International Workshop on Information Security Applications*. Springer, 27–37.

[78] Ubaidullah Rajput, Fizza Abbas, and Heekuck Oh. 2018. A solution towards eliminating transaction malleability in bitcoin. *J. Inf. Process. Syst.* 14, 4 (2018), 837–850.

[79] P. Rakdej, N. Janpitak, M. Warasart, and W. Lilakiatsakun. 2019. Coin recovery from inaccessible cryptocurrency wallet using unspent transaction output. *Proceedings of the 4th International Conference on Information Technology: Encompassing Intelligent Technology and Innovation Towards the New Era of Human Life (InCIT 2019)*, 99–103.

[80] Fergal Reid and Martin Harrigan. 2013. An analysis of anonymity in the bitcoin system. In *Security and Privacy in Social Networks*. Springer, 197–223.

[81] H. Rezaeighaleh and C. C. Zou. 2019. New secure approach to backup cryptocurrency wallets. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'19)*. 1–6. https://doi.org/10.1109/GLOBECOM38437.2019.9014007

[82] H. Rezaeighaleh and C. C. Zou. 2020. Multilayered defense-in-depth architecture for cryptocurrency wallet. In *Proceedings of the IEEE 6th International Conference on Computer and Communications (ICCC'20)*, 2212–2217. https://doi.org/10.1109/ICCC51575.2020.9345013

[83] A. R. Sai, J. Buckley, and A. Le Gear. 2019. Privacy and security analysis of cryptocurrency mobile applications. In *Proceedings of the 5th Conference on Mobile and Secure Services (MobiSecServ'19)*. 1–6. https://doi.org/10.1109/MOBISECSERV.2019.8686583

[84] R. Soltani, U.T. Nguyen, and A. An. 2019. Practical key recovery model for self-sovereign identity based digital wallets. In *Proceedings of the IEEE 17th International Conference on Dependable, Autonomic and Secure Computing, IEEE 17th International Conference on Pervasive Intelligence and Computing, IEEE 5th International Conference on Cloud and Big Data Computing, 4th Cyber Science and Technology Congress (DASC-PiCom-CBDCom-CyberSciTech'19)*, 320–325.

[85] Aakanksha Soni and Saurabh Maheshwari. 2018. A survey of attacks on the bitcoin system. In *Proceedings of the IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS'18)*. IEEE, 1–5.

[86] S. Suratkar, M. Shirole, and S. Bhirud. 2020. Cryptocurrency wallet: A review. In *Proceedings of the 4th International Conference on Computer, Communication and Signal Processing (ICCCSP'20)*. https://doi.org/10.1109/ICCCSP49186.2020.9315193

[87] H. Takahashi and U. Lakhani. 2019. Multiple layered security analyses method for cryptocurrency exchange servicers. In *Proceedings of the IEEE 8th Global Conference on Consumer Electronics (GCCE'19)*, 71–73.

[88] F. Teomete Yalabik and I. Yalabik. 2019. Anonymous bitcoin v enforcement law. *Int. Rev. Law Comput. Technol.* 33, 1 (2019), 34–52.

[89] Philipp Tschannen and Ali Ahmed. 2020. On the evaluation of the security usability of bitcoin's APIs. In *Proceedings of the Evaluation and Assessment in Software Engineering*, 405–412.

[90] Md Shahab Uddin, Mohammad Mannan, and Amr Youssef. 2021. Horus: A security assessment framework for android crypto wallets. In *International Conference on Security and Privacy in Communication Systems*. Springer, 120–139.

[91] E. Ulqinaku, J. Stefa, and A. Mei. 2019. Scan-and-pay on android is dangerous. In *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'19)*.

[92] L. Van Der Horst, K.-K. R. Choo, and N.-A. Le-Khac. 2017. Process memory investigation of the bitcoin clients electrum and bitcoin core. *IEEE Access* 5 (2017), 22385–22398. https://doi.org/10.1109/ACCESS.2017.2759766

[93] Helen Mary Varghese, Dhwani Apurva Nagoree, N. Jayapandian, et al. 2021. Cryptocurrency security and privacy issues: A research perspective. In *Proceedings of the 6th International Conference on Communication and Electronics Systems (ICCES'21)*. IEEE, 902–907.

[94] M. Vasek, J. Bonneau, R. Castellucci, C. Keith, and T. Moore. 2017. The bitcoin brain drain: Examining the use and abuse of bitcoin brain wallets. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9603, 609–618. https://doi.org/10.1007/978-3-662-54970-4_36

[95] Tejaswi Volety, Shalabh Saini, Thomas McGhin, Charles Zhechao Liu, and Kim-Kwang Raymond Choo. 2019. Cracking bitcoin wallets: I want what you have in the wallets. *Fut. Gener. Comput. Syst.* 91 (2019), 136–143. https://doi.org/10.1016/j.future.2018.08.029

[96] E. Zaghloul, T. Li, M. W. Mutka, and J. Ren. 2020. Bitcoin and blockchain: Security and privacy. *IEEE IoT J.* 7, 10 (2020), 10288–10313. https://doi.org/10.1109/JIOT.2020.3004273

[97] Junhuan Zhang, Yuqian Xu, and Daniel Houser. 2021. Vulnerability of scale-free cryptocurrency networks to double-spending attacks. *Eur. J. Financ.* (2021), 1–15.

[98] Y. Zhang, M. Luo, K.-K.R. Choo, L. Li, and D. He. 2020. Efficient and secure two-party distributed signing protocol for the gost signature algorithm. *Commun. Comput. Inf. Sci.* 1298 (2020), 3–19.

[99] Fangdong Zhu, Wen Chen, Yunpeng Wang, Ping Lin, Tao Li, Xiaochun Cao, and Long Yuan. 2017. Trust your wallet: A new online wallet architecture for bitcoin. In *Proceedings of the International Conference on Progress in Informatics and Computing (PIC'17)*. IEEE, 307–311.

[100] S. Zollner, K.-K. R. Choo, and N.-A. Le-Khac. 2019. An automated live forensic and postmortem analysis tool for bitcoin on windows systems. *IEEE Access* 7 (2019), 158250–158263. https://doi.org/10.1109/ACCESS.2019.2948774

Received 28 March 2022; revised 25 January 2023; accepted 2 May 2023