OST
Eastern Switzerland
University of Applied Sciences

INS | Institute for
Network and Security

EASTERN SWITZERLAND
UNIVERSITY OF APPLIED SCIENCES

FOCUS PROJECT 1

A Security Focused Outline on Bitcoin Wallets

Author:
Roman BÖGLI

Supervisor:
Prof. Dr. Nathalie WEILER

A work submitted in fulfillment of the requirements for the degree of Master of Science in Engineering in Computer Science (MSE CS)

at

Institute for Network and Security (INS),
Departement of Computer Science

7. March 2023

Block Height 779'710

Abstract

The famous electronic peer-to-peer cash system called Bitcoin is an open-source protocol allowing individuals to store and transact units of the same named currency. Private and public key cryptography plays a central role in this value transfer system, which implies the importance of professionally managing the information about such keys.

This work elaborates on the essential prerequisites to understand this relatively new technology that combines elements from the fields of computer science, cryptography, mathematics, and game theory. In doing so, crucial general and Bitcoin-specific terms are defined and contextually explained.

The central part of this work addresses the outline of different Bitcoin interaction means, commonly known as wallets. The structure of the presented wallet types orients itself alongside a potential user's experience. Besides defining explanations and examples of use cases, this work outlines advantages and disadvantages concerning security and privacy.

The start concerns two wallets that target beginners in the field of Bitcoin. The concept of online accounts is elaborated and attention is drawn to the inherent need to trust when using them. Also, the relatively primitive type of paper wallets is surveyed.

For a more intermediate interaction with this peer-to-peer cash system, the concept of software wallets, in general, is explained and examples are provided. The bridge from single-address paper wallets will be drawn to the more sophisticated multi-address wallets enabled through rooted key derivation techniques. Designated computer devices that solely serve the purpose of managing keying material, known as hardware wallets, represent another intermediate wallet type discussed in this work.

Last, advanced topics are discussed that further leverage the security and privacy of someone's interaction with Bitcoin. One concerns the setup of a self-managed Bitcoin full node. This undertaking not only harmonies with the concept of verification over trust but also allows for the complete exclusion of any third party between wallet communication. Equally advanced is the concept of multi-signature wallets, which is discussed at the end of this work.

Keywords: Bitcoin, Software Wallets, Hardware Wallets, Private Key Management

Declaration of Authorship

I, Roman BÖGLI, declare that all material presented in this paper is my own work or fully and specifically acknowledged wherever adapted from other sources. I understand that if at any time it is shown that I have significantly misrepresented material presented here, any degree or credits awarded to me on the basis of that material may be revoked. I declare that all statements and information contained herein are true, correct and accurate to the best of my knowledge and belief.

Contents

1 Introduction

In October 2008 a user with the pseudonym Satoshi Nakomoto introduced in an email to *The Cryptography Mailing List* the idea of a peer-to-peer electronic cash system that no longer requires trusted parties and simply works based on software and mathematical rules [1]. The roots of such a digital cash system reach back to the year 1983 [2]. Nakamoto referenced a PDF in this email that explains this idea in detail alongside proof of its robustness. This document later became famous and is nowadays also referenced as the so-called *Bitcoin White Paper* [3].

At the beginning of 2009, Nakamoto published a post in the *P2P foundation forum* to invite the public to explore and download the first software version of Bitcoin [4]. Shortly after this announcement, the idea was discussed and further developed by the back at this time a small group of people that followed the mailing list or the activities in this forum. Also, the meanwhile famous Bitcoin logo was created, as shown in Figure 1.1.

Looking back at this early stage in the course of Bitcoin's evolution, these posts represent contemporary history[1]. More than ten years later, Bitcoin still exists and continues to be collaboratively improved by thousands of people. In the meantime, it also served as a source of inspiration for new business models, technological devices, and how people transact with each other.

Nakamoto's peer-to-peer electronic coin that can be used to transfer units from one place to another without relying on a necessarily trusted party in between succeeded. Numerous individuals and even institutions benefit from it by using it as a store of value, medium of exchange, or even unit of account. These purposes are commonly known as the three purposes of money.

An idea, textual explanation, software source code, mathematical laws, and intrinsic interest of individuals form the basis of this success. One concludes that everything

FIGURE 1.1: Bitcoin Logo
The logo was created collaboratively by a few Bitcoin forum users starting in February 2010. [5]. Usually, the upper case spelling «**B**itcoin» denotes the protocol and the related technology as well as community while the lower case spelling «**b**itcoin» denominates the currency unit that is also abbreviated using `BTC`. For the sake of consistency, only the upper case variation and its abbreviation is used in this work.

[1] See the browsable collections of it.

that defines Bitcoin is simply *information*. Apparently, Nakamoto intended to free this information to the world and therewith pass the point of no return. Everyone and everything that can process this information will be able to interact with Bitcoin. No permission has to be obtained, nobody and nothing needs to be trusted, nor could by any authority.

It must have been Nakamoto's desire to enable such a complete self-determined interaction for absolutely everyone without any prevailing party involved. This may be why the pseudonymized authorship has not been revealed until today and possibly never will. A potential association of Bitcoin with any form of existing individual or collective would automatically increase their influential power, regardless of whether indented. Additionally, this would also represent a target for powerful institutions and governments that may repudiate the idea of an uncontrollable, transparent, and censorship-free value transfer system. Ultimately, it is of no meaning who published this idea. What is essential, however, is that millions of people absorbed it and continue to use it because they intrinsically want to use it.

This work should lower the admittedly high barrier to entry to Bitcoin, again increasing its accessibility. This is done by covering different ways to interact with this peer-to-peer system with different security and privacy levels. While aiming for practicality, a generic storyline will be narrated where the readership identifies itself. The story told reflects the questions and challenges a new Bitcoin user will sooner or later be faced with. This undertaking structures in three stages, namely beginner, intermediate, and advanced. Eventually, the readership has gained the necessary information to confidentially start interacting with Bitcoin using the approaches that best fit the user's needs.

The present work does not address money's historical, social, and economic aspects and its relation with Bitcoin. Interested readers in these topics can refer to Von Mises [6] and Ammous [7] for a start. In this work, the focus merely lies on the proper management of the private information required to interact with Bitcoin and the prerequisites necessary to understand the risks and benefits of the presented approaches.

1.1 Motivation

Today's societies and industries rely heavily on *Information and Communication Technologies (ICT)*. Data is the new oil, people say. Everything is or is going to be connected, machines communicate with each other and humans start to spend their spare time in vitalized worlds. ICT became a backbone in both society and industry. Its omnipresence in everyday life is indisputable, which is why awareness and education in this field became inevitable.

However, there is another field that is equally omnipresent, namely financial transactions. Nearly no day passes without taping a credit card or handing over cash to pay the counterpart and therewith complete a transaction of goods or services between peers.

The convergence between technological progress in information systems and the indispensable demand for financial transactions affects everybody who wants to take part in the commercialized environment, which is, in fact, hard to escape. Although people are indirectly forced to participate in the system of digital payments, they have little power over how, when, and where their financial data points are processed, stored, and benefited from. Identities that control or offer financial services, such as governments, banks, or payment processors, find themselves in the fortunate situation that their user base must rely on and trust them. This asymmetric relationship compromises financial autonomy, privacy, and liberty.

Bitcoin offers an alternative to this custodianship. It represents a value transfer system that no longer enforces an asymmetric relationship between provider and user simply because there is no provider anymore. It serves as an inflation-protected store of value, a private medium of exchange, and an opportunity to have an absolute unit of measurement. Everyone and everything can unconditionally participate in this system in any desirable form and will be able to successfully interact with it as long as protocol rules, which are identical for all users, are adhered to.

This freedom, however, comes with the cost of self-responsibility. Not having a provider or centralized authority also means that no customer helpline can be consulted. Of course, there exist institutions or individuals that provide support services to different extents. Starting with personal explanatory sessions and reading material recommendations and ending with the complete management of one's Bitcoins. The latter solution to the risk of self-responsibility, which by then represents a more or less wholly mitigated risk, compromises the benefits of Bitcoin again and should thus be undesirable.

Managing the risk of self-responsibility requires further knowledge about Bitcoin. It distills itself into proper and secure management of private information, i.e., information that should never be disclosed to any other person or party unless the one who should know about it. This work aims to provide this knowledge in a step-wise manner. Firstly, it defines what information a Bitcoin user requires to handle. This is followed by different techniques specifying how to handle this information securely. The techniques presented in this work differ from each other as they become more and more sophisticated in terms of security and privacy, but also at the cost of higher complexity. It should be the reader's choice at what stage the work serves its purpose.

1.2 Free and Open-Source

Virtual assets such as photographs, songs, or source code possess the common property of effortless duplication and distribution since they represent information encoded in zeros and ones. This significant difference to physical assets requires means to control the technically unrestricted exploitation of virtual goods in the digital age. One protection technique would be to keep a virtual asset secret or only publish parts of it. Another and often more helpful strategy involves licenses as software that embodies original work

generally is protected by copyright[2]. Licenses empower authors to decide how, when, where, and by whom their virtual assets may be used. This, for instance, has become an essential tool in the field of commercialized software products.

Software that does not ought to be commercialized is often gifted to the public domain by publishing the entire source code to the world for everybody to see and use. The yearly report by Linux & Open Source Annual [8] provides a comprehensive overview of the extent and diversity of such software. Such software is then referred to as *free* and/or *open-source*. Although the two terms describe the same effects in the context of software, they nevertheless associate slightly different values with it. Williams and Stallman [9] and Haff [10] discuss these values in detail alongside their history and background information. The Open Source Initiative [11] has defined its term using ten basic concepts that must be fulfilled to call a piece of software open-source legitimately. The ideology of these concepts focuses on objective facts related to program code. Many different open-source licensing models exist that can specify software usage, manipulation, or distribution to different extents [12].

Free software reaches further into a philosophical dimension. According to Williams and Stallman software can be entitled as free if and only if its users possess the freedom to study, run, modify, and share its source code. In that sense, the authors affiliate the mnemonic *«free speech, not free beer»* to emphasize that it is all about freedom, not price. Stallman is seen as an activist for this movement and has contributed to it through valuable undertakings[3]. Free software implies it is open-source while open-source software must not necessarily allow for the required freedom to its users in order to be called free software. Often the two terms are also combined such as *Free and Open-Source Software (FOSS)* or even *Free/Libre and Open-Source Software (FLOSS)* to increase clarity.

FOSS forms one of the most essential pillars of Bitcoin. The reference implementation of the Bitcoin protocol known as *Bitcoin Core* stays at everyone's disposal and can freely be downloaded from the internet[4].

Allowing for maximum transparency regarding its inner functioning makes Bitcoin *trustless*. No entity can practically prohibit access to the source code, much less any form of its usage which makes Bitcoin *permissionless*. Information in the form of source code spreads effortlessly over the internet, utterly unregarded of any geographic background which makes Bitcoin *borderless*. Having access to everything required to operate a personal instance autonomously in a personally chosen environment makes Bitcoin *decentralized*. These characteristics together make Bitcoin *censorship-resistant* as no authority can practically restrict or prevent the interaction possibilities with Bitcoin[5]. Eventually, this allows

[2] For example in Switzerland defined by the Copyright Act (CopA) Art. 2, Section 3.
[3] Richard Stallman founded the Free Software Foundation (FSF) and the GNU General Public License.
[4] For example from its GitHub repository under MIT license, which grants everyone *«the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software»* as long this license notice remains associated.
[5] Current events in the Lebanese banking sector, for instance, exemplifies the value of these properties.

the emergence of a user base formed by self-determined individuals that independently from each other are empowered to transact with each other on the Bitcoin network.

The commercial domain has witnessed the benefits arising from the FOSS methodology in recent years. Gartner's 2021 Hype Cycle report [13] represents this phenomenon. Open-source software in the domain of office programs or data storage is expected to reach the plateau of productivity in five to ten years, and open-source technology in the field of natural language processing even in under five years. Big technology companies such as Microsoft or Apple[6], for instance, started disclosing their code repositories. They keep fostering the growth of communities around them which is in turn rewarded by positive publicity. Ultimately one can state that the trend toward transparency and self-determination in software code is genuine and likely to augment in the future.

1.3 Other Resources

The work is not the first of its kind nor will it probably be the last. The *National Institute of Standards and Technology (NIST)* published a comprehensive tripartite report concerning the subject of cryptographic key management. Part 1 [14] elaborates on the basics by providing definitions, best practices in handling keying material, and discussing acknowledgeable pitfalls. Part 2 [15] provides requirements and recommendations regarding cryptography policies and governance to succeed with institutional key management. Part 3 [16] addresses the proper management of keying material using available applications, protocols, and infrastructures. Eskandari et al. [17] discuss the challenges that evolve from the inevitable involvement of public key cryptography in Bitcoin and assess the usability and security of different key management techniques for Bitcoin.

1.4 Outline

The rest of this document is structured as follows. Chapter 2 is dedicated to the technical prerequisites. It explains the basic functionalities of Bitcoin, defines essential terms, and references more in-depth resources on the various technical components Bitcoin builds on. Chapter 3 discusses the basic Bitcoin management tools that new users usually are exposed to when they start informing themselves about this domain. Chapter 4 introduces more versatile tools that allow for more sophisticated Bitcoin interactions while profiting from better privacy and security. Chapter 5 discusses more advanced topics to improve further the privacy of Bitcoin users and a special type of wallet will be elaborated. Last but not least, Chapter 6 will conclude the presented insights and provide an outlook on further work.

[6] See Microsoft's FOSS ecosystem or collection of Apple's open-source projects.

2 Prerequisites

This chapter defines and explains the essential elements directly or indirectly connected to Bitcoin, thus required to comprehend the rest of this work. Understandably, not all used expressions that belong to the taxonomy of ICT will be discussed in this chapter. Although a basic acquaintance with the computer science field will help understand the discussed subjects in this work, it also addresses the readership with different backgrounds.

The provided explanations aim for shortness and conciseness as this work primarily focuses on the actual interaction with Bitcoin. Resources to more detailed resources will be provided.

2.1 General Terms

This section defines the most critical general terms found in the field of computer science and cryptography. All of them existed far before the introduction of Bitcoin and are considered crucial tools in many other fields of applications. Easttom [18], Aumasson and Green [19], and Aumasson [20] serve as a basis for the following definition and represent recommended resources for further information.

2.1.1 Protocol

Generally speaking, a protocol can be seen as a set of rules or expected behavior applicable in some situations. In computing, these situations concern the communication of connected computers. When the only form of communication is bound to zeros and ones, it becomes evident that a detailed rule set is required to succeed. Internet users constantly and probably unconsciously profit from several protocols. The *Hypertext Transfer Protocol/Secure (HTTP/S)* is probably the most prominent one among them.

2.1.2 Network

A network generically describes the situation of several independent entities that share some connection to each other. A single person's relationships, for instance, form their social network. Likewise, computers can be interconnected to form a digital network. Whether humans or computers, networks purpose to share information through communication. This, on the other hand, requires that all participants in a given network adhere to a common protocol, as described previously.

2.1.3 Hash Function

Equally essential as protocols are hash functions. Every computer user indirectly relies on them without probably knowing it. Although there exists many different types[1] of hash functions, this prerequisite focuses on so-called *cryptographic hash functions*. For instance, the *Secure Hash Algorithm (SHA)* family represents a prominent set of such hash functions and is based on the *Secure Hash Standard (SHS)* defined by the NIST [21].

One can generally define a hash function as a mapping $h : \{0,1\}^{\star} \mapsto \{0,1\}^{n}$ whereas n represents a constant defined in the respective implementation. In other words, hash functions take binary information of arbitrary length $\star$ as input and return binary information of fixed length n as an output. This output is known as *hash value* or *digest*.

Such a mapping from an infinitely large space to a finite space inevitably leads to so-called *collisions*, i.e. two different inputs result in the same output[2]. When the frequency of collisions resides at an acceptable level, one considers the hash function secure. In other words, the probability that two different inputs result in the same hash value is acceptable small. Consequently, secure hash functions represent a powerful tool to protect data integrity as modifications in a specific input can immediately be detected by comparing the resulting hash values. Aumasson and Green [19, Chapter 6] discuss the properties of secure hash functions in further detail.

It is furthermore notable that cryptographic hash functions exhibit four unique properties. Firstly, they must be *deterministic*, meaning that the same input repetitively leads to the same output. Secondly, they should be *unpredictable* so that no assumptions about a potential output can be inferred by analyzing a given input. Furthermore, they should also be *irreversible*, i.e. one-way functions, creating the impossibility of calculating the input from a given output. Lastly, they should be *chaotic*, meaning that little change in the input results in a significant change in the output.

Cryptographic hash functions are nowadays used, for example, in password authentication schemata where servers only receive the hash values of the user's secret passwords. Another use case involves the generation of so-called *fingerprints* for large amounts of data. It is common practice to rely on prominent hash function implementations[3] rather then inventing its own. In addition to passwords and numerous other fields of ICT application, cryptographic hash functions also play a central role in Bitcoin, as will be shown later.

[1] See list of categorized hash functions.
[2] Also known as pigeonhole principle.
[3] For example `MD5` or the `SHA-2` family.

2.1.4 Public-Key Cryptography

Other than symmetric cryptography where the same piece of private information, denoted as *key*, is used for encryption and decryption, public-key cryptography provides two different keys for these fundamental operations. One therefore also refers to it as *asymmetric* cryptography. One key is referenced as the so-called *public key* and can openly be communicated. The other key is called a *private key* and must always be kept secret. Although they visually appear to be completely different, both keys relate mathematically to each other.

The following example demonstrates the roles of these keys. Bob uses an asymmetric cryptography algorithm to create a new key pair consisting of a private key K and a public key P. Bob publishes P on his webpage so that everyone interested can read it. When Alice wants to send a secret message to Bob over an insecure channel, she utilizes Bob's public key P to encrypt her message M which results in the cipher text C. Alice can send C over a public and possibly insecure channel to Bob who subsequently decrypts C using his private key K and read the plain text M.

Popular implementations for algorithms that utilize such public-key cryptography are for example *Rivest-Shamir-Adleman (RSA)*, *Diffie-Hellman Key-Exchange (DHKE)*, or *Elliptic Curve Digital Signature Algorithm (ECDSA)*. Das and Madhavan [22, Chapter 5] surveys these in further details.

2.1.5 Digital Signatures

Public-key cryptography can also be used to prove the authenticity of digital information through signatures. The distribution of source code represents a famous use case for digital signatures as it helps to answer the question of whether the downloaded code is identical to the author's indented version. Therefore, authors like Alice could encrypt the code base B or its digest using her private key K resulting in a digital signature S_d. Usually, S_d is then published alongside B. After B and S_d were downloaded, users decrypt S_d using Alice's publicly known public key P and verify the authentic result with the received version of B resp. its digest. In other words, digital signatures provide a powerful tool to verify data authenticity.

Besides this, two additional functionalities are (theoretically) accounted for when applying digital signatures during data transfers. One is the origin authentication, meaning the assurance from whom the information package was sent. The other denotes the fact that the signer cannot reasonably deny a signature, or more specifically, the knowledge of the private key, which is also known as *non-repudiation*. However, the latter two properties only remain meaningful when the signer's secret private keys are never disclosed. If this happens anyway, every third party could have created the signature in an equally valid form.

Encoding	Value
ASCII	Satoshi
Binary	01010011 01100001 01110100 01101111 01110011 01101000 01101001
Base16	5361746F736869
Base64	U2F0b3NoaQ==

TABLE 2.1: Different Encodings of the Word «Satoshi»
This table examplifies the different ways to encode information. The ASCII characters forming the word «Satoshi» can translated into utterly different forms.

2.1.6 Encodings

Digital information in its most basic form of zeros and ones is, however, deficient in convenience for human comprehension. Thus one created standardized translations of binary code using different character sets, which is in this context known as *binary-to-text-encoding*. There exists a wide range of encodings[4] with ASCII[5], Base16 (also known as *hexadecimal*), or Base64 among the most prominent ones. Table 2.1 illustrates this with the example of the word «Satoshi». Although the values differ significantly in length and character set, they still embody the same information.

2.1.7 Entropy

Entropy belongs to the fundamental concepts in the field of information theory and was formally defined by Shannon [23] in 1948. Simplified, one can state that entropy represents a unit of measurement used to declare the amount of information content present in a given system, for example, a digital message. The message's entropy can be derived by counting the required bits to transport the information, neglecting redundancy.

Alternatively, entropy also defines a measurement of uncertainty. Although uncertainty and information seem to be contradictive concepts, Shannon reasons them as equivalent. A message that includes something already known by the receiver can, technically speaking, not be considered information. Consequently, only unknown or uncertain things bear information content. This leads to the conclusion that uncertainty *is* information.

Equation 2.1 generally formulates how to calculate entropy. It expresses the entropy of a system x as the negative sum of all probabilities p_i that exist among the probability distribution of x multiplied with their binary logarithm[6]. Since $\log_2 n$ with $n \in [0, 1]$ yields negative intermediate results, the negative summation will ensure a positive entropy value.

[4] See most used binary-to-text encoding standards.
[5] Abbreviation for *American Standard Code for Information Interchange (ASCII)*.
[6] Also known as «logarithm base two» ($\log_2$).

$$E(x) = -\sum_{i=1}^{n} p_i * \log_2 (p_i) \tag{2.1}$$

In other words, when every bit in a sequence of bits exhibits high uncertainty regarding its binary value, it features high entropy. The entropy would decline if certain binary states were more likely than others as it consequently contains less uncertainty. Password generators aim for high entropy as it qualifies the strength of a chosen password. Easttom [18, Chapter 3] further discusses entropy as part of the core concepts in cryptography.

2.2 Bitcoin Terms

After specifying some required general concepts encountered in the world of computers, this section addressed the most crucial terminology specifically used in Bitcoin. Certain terms are most probably already encountered by the readership thanks to the medial presence of Bitcoin in recent years. However, a clear understanding of these terms often needs to be fulfilled due to their novelty and technological complexity. This chapter, therefore, addresses this deficit by providing definitions and further explanations of the most crucial components in Bitcoin based on the works by Antonopoulos [24], Antonopoulos, Osuntokun, and Pickhardt [25], and Van Oorschot [26].

2.2.1 Node

Many computers connected and capable of consent on a common form of communication define a network, as previously stated. This is no different in Bitcoin. Every computer device with an active internet connection can download, install and run the source code of any Bitcoin client, i.e. software that implements the Bitcoin communication protocol alongside its standards. They thereby automatically become part of the network as a so-called *node*.

Usually, one distinguished between *full* and *light* or *lightweight* node with the difference being that a full node permanently stores a copy of the current state of all important information available in the network while light nodes depend on other full nodes and/or prune the stored information. Consequently, light nodes rely on the blockchain replicate of some other party which requires trust. Full nodes on the other hand create, verify, and manage their own copy, making them completely independent from any other information sources.

Another differentiating characteristic is associated with the node's ownership. So-called *self-hosted* or *domestic* nodes represent fully controlled nodes by the hosting entity. Consequently, existing nodes that any third party controls are denoted as *foreign* nodes in this work.

It is essential to understand that nodes are not constrained by any geographical much less political boundaries. Technically one could also develop its own version of a Bitcoin node and successfully use it to interact with the network as long as the consented protocol remains respected. Receives as protocol-adhering node N_a any protocol-divergent communication from another node N_d, so will N_a automatically eliminate its connection to N_d as a consequence. This will eventually result in the complete detachment of N_d to the network as other adhering nodes behave identically.

2.2.2 Private & Public Key

The concept of private and public keys in terms of asymmetric cryptography schemata was discussed in the previous section. The reason why this subject is revisited in this section lies in the fact that these keys represent one of the most central elements in the course of understanding Bitcoin.

A private key K is merely a randomly chosen 256-bit number in the interval $[1, n-1]$ with $n = 1.158 * 10^{77}$ resulting in slightly fewer numbers to choose from than 2^{256}. The reasoning behind this lies in the prevalent mechanism to generate keys in Bitcoin, namely the ECDSA. The chosen elliptic curve in Bitcoin reasons for this specific upper bound of $n-1$. Antonopoulos [24, Chapter 4] elucidates the inner working of the ECDSA in further detail. It can be challenging for humans to comprehend the vast number of eligible private keys resulting from this given range. In fact, there are about as many possible private keys as there are atoms in the observable universe [27].

Next, a constant G is used to multiply K in order to receive the associated public key $P = G * K$. This operation is referred to as *Elliptic Curve Multiplication* and represents a cryptographic one-way function. In other words, deriving P from K using G is easy while the attempt to restore K from a given P is practically impossible as it embodies the so-called *Discrete Logarithm Problem (DLP)*[7]. In simplified terms, this problem describes a trivial mathematical procedure to compute in one direction but is extremely difficult to inverse.

Meanwhile, other mechanisms were included in the Bitcoin standard to generate these key pairs. The most recent one, for example, is known as *Taproot* and utilizes the digital signature scheme by Schnorr [28] for the key pair generation instead of the described ECDSA.

Table 2.2 summarizes the existing address standards to this date. In the beginning, there existed only *Pay to Public Key Hash (P2PKH)* address i.e. the most basic lock script. Later *Pay to Script Hash (P2SH)* and *Pay to Witness Public Key Hash (P2WPKH)* were introduced, which allow for more complex scripts as well as more storage-efficient structures. The most recent address type *Pay to Taproot (P2TR)* further expands the scripting possibilities while offering increased privacy.

[7] Finding x for a given b in $a^x \equiv b \mod p$ is disproportionately harder than finding b for a given x.

Type	Prefix	Description
Pay to Public Key Hash (P2PKH)	`1...`	First address type, represents the hash value of a public key using `Base58Check` encoding, allows only for private key proof as spending condition, nowadays considered as legacy.
Pay to Script Hash (P2SH)	`3...`	Successor of P2PKH through BIP16, represents the hash value of a script that can embody more complex spending conditions, yield for optimized transaction compression using *Segregated Witness (SegWit)*.
Pay to Witness Public Key Hash (P2WPKH)	`bc1q...`	Successor of P2SH through BIP141 and BIP142, also known as *Native SegWit*, further reduces transaction size, uses `Bech32` encoding, considered the most popular address type at the moment.
Pay to Taproot (P2TR)	`bc1p...`	Latest address standard (BIP341), further reduces transaction sizes compared to P2SH for complex scripts, further widens the scripting possibilities through its new Merkle tree structure, improves privacy through Schnorr signatures (BIP340).

TABLE 2.2: Bitcoin Address Standards
Shows the various standardized address types that exists in Bitcoin. The standards were defined in such a way that the address types can be differentiated by looking at the addresse's prefix.

Alongside the ongoing development of the Bitcoin protocol, these address standards were introduced to further optimize transaction attributes such as storage size, functionality, or privacy. Each type has a standardized encoding resulting in address formats that allow identification using the address prefix.

It is worth mentioning that the cryptographic hash function applied in the address generation procedure maps the domain of feasible public keys to a slightly smaller range of possible hash values. Consequently, multiple public keys could technically lead to identical addresses. Since the amount of possible public keys resides in such an enormously ample number space, however, this is not considered to be an issue in practice.

Another interesting fact about Bitcoin address encodings is that they are optimized for human interaction. For example, they do not include look-alike characters (e.g. zero and «O» or «L» and «i») to improve readability. Additionally, checksum mechanisms allow for detecting typing errors to a certain extent. Although such counter measurement for human error exists, it is recommended to rely on computer-aided means for address communication such as copy-pasting digital characters or with the help of *Quick Response (QR)* codes.

2.2.3 Unspent Transaction Outputs

When people refer to any action concerning Bitcoin they colloquially use verbs such as «owing», «storing», or «sending» any amount of this virtual currency. Technically, this is a misleading enunciation. However, what is an actual chunk of a Bitcoin unit is a so-called *Unspent Transaction Output (UTXO)*. Every full node maintains and constantly updates its copy of the current UTXO set, i.e. all available UTXOs in the Bitcoin network.

Another way to look at this set is by imagining it represents all spendable coins. Simplified explained, does an UTXO represent a credit that is cryptographically bound to a specific public key, respectively its hash value known as Bitcoin address. Since binding UTXOs to public key address through P2PKH operations was originally the only use case in that sense, this binding was referred to as `scriptPubKey`. In the meantime, more complex types of imposing a spending condition evolved, which established the more generic names *lock script*, *witness script*, or simply *cryptographic puzzle*.

In order to successfully spend or consume an UTXO, the underlying puzzle or lock script must be solved resp. unlocked. The only way to accomplish this is by successfully fulfilling the underlying locking condition. Initially, these fulfillments were mostly accomplished using digital signatures, which can only be generated by the entity that possesses the compatible private key. One therefore originally referred to it as `scriptSig`. As other forms than signatures exist to fulfill a locking condition, the more general names *witness* or simply *unlocking script* evolved.

To summarize, units of Bitcoins only exist in the form of lock scripts embodied in UTXOs and transferring them requires the presentation of a proper unlocking script which primarily includes the signatures created from the appropriate secret private keys. Consequently, Bitcoins are not directly «owned». What can be owned, however, are private keys which in turn can unlock the lock script in UTXOs and by doing so moving Bitcoins from one virtual place to another, i.e. locking them using new conditions after the have been unlocked. It follows that the private keys actually represent the possessable chunks of Bitcoins, which has established the saying *«not your keys, not your coins»*.

2.2.4 Transactions

As previously learned, transferring Bitcoins and by doing so essentially creating financial transactions is nothing less than reallocating spendable UTXOs by unlocking the associated lock script. The P2PKH transaction essentially enables the act of spending an UTXO by transferring it to another public key address. More complex transaction types allow for lock scripts that only can be unlocked after a certain time passed and/or more than one matching signature is provided. Bistarelli, Mercanti, and Santini [29] survey these advanced transaction types.

Real coins serve as an excellent analogy to understand the general structure of Bitcoin transactions. When Alice wants to buy a $3 coffee, she will look into her wallet and collect

any amount of coins that results in precisely this price or the closest approximation to it. In the latter case, the payee will return a change to Alice.

Now imagine Alice possesses a private key that controls two UTXOs denoted with 0.4 and 0.3 Bitcoin (BTC). When Alice now wants to send BTC 0.6 to Bob, she must consume these two UTXOs and use them as so-called *transaction inputs*. Consuming in this context means that Alice will create an unlocking script for both of her UTXOs that fulfills the spending condition embodied in respective locking scripts. It follows that every transaction input contains both a locking and an unlocking script. Next, a new UTXO with the amount of 0.6 is created that will only be spendable by the owner of the private key to the public address Bob gave to Alice. In other words, Alice creates a new UTXO by incorporating Bob's public key address in the associated lock script. Since the sum of all inputs equals 0.7 but only 0.6 should leave Alice's wealth to Bob, she is responsible for adding another transaction output of maximum 0.1 units. However, the script of this second output will be locked using Alice's public key address as it represents her change and should remain spendable.

One can conclude that a payer places sufficient spendable UTXOs as a transaction input and creates the desired new UTXOs that account for the payee's claim and, if required, the payer's change. Figure 2.1 models this constant mapping of any spendable inputs to any outputs, given that enough funds are provided on the input side in order to cover the output side.

In practice, the summation of all values from the input side usually exceeds the output side. The implicit delta is considered the *transaction fee* and will be credited to the entity that settles this transaction. The following two chapters will address the transaction settlement process in detail. Antonopoulos [24, Chapter 6] elaborates on the architecture of transactions and the associated central role of secure digital signatures in great detail.

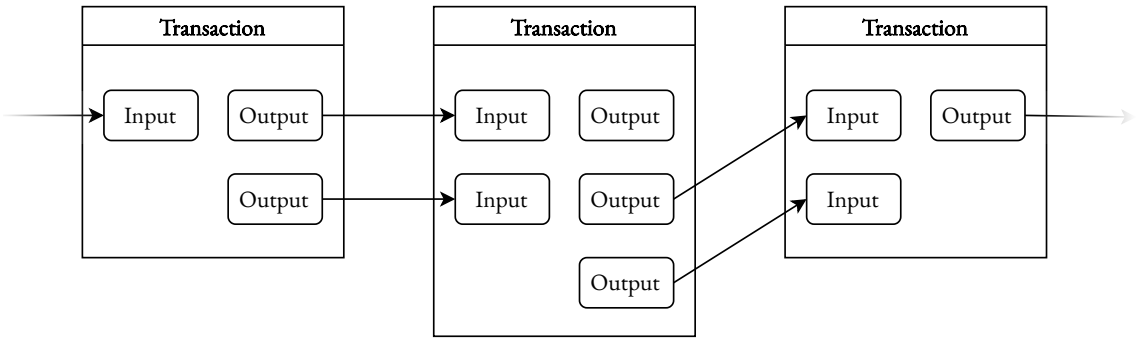

FIGURE 2.1: Transaction Inputs & Outputs
Bitcoins are transacted by mapping spendable inputs (i.e. the creator's UTXOs) to any number of outputs. The number of inputs, outputs, or its ratio is unconstrained as long as sufficient funds are provided on the input side.

2.2.5 Blockchain

Up to this point, it was shown how individual entities or nodes can independently create a communication network through open-source software that implements specific protocol rules. What still needs to be added so far is the component that stores the knowledge about passed communication i.e. transactions in this network. Banks and payment processors rely on centralized databases to correctly determine the current balance of someone's currency account. Different time zones, operating hours, internet connectivity states, or payment terminals convert this into a difficult task which is why, for instance, credit card payments remain pending for several business days.

In Bitcoin, the synchronization of transactional knowledge is accomplished by means of a data structure conventionally known as *blockchain*. It describes a concept where transactions are chronologically bundled into small data package alongside with a unforgeable reference to its preceding package. More concretely, this reference is the hash value of the entire previous data package.

Figure 2.2 visualized this process. The hash value of block $[n-1]$ together with the entire block $[n]$ serve as input for the hash function returning a new block hash value which subsequently will serve as input for the next block. Thus, Bitcoin's blockchain represents a universal *timechain* to synchronize knowledge worldwide chronologically.

Every full node in the network maintains its own copy of the entire blockchain. This allows for autonomous verification of every single transaction's locking and unlocking scripts, making trusted third parties obsolete.

On the other hand, the complete transparency regarding every single transaction that has ever been processed bears the costs of reduced anonymity. Anonymity protection happens exclusively with the unknown linkage between a specific set of addresses and a user's real-world identity. Therefore, Bitcoin is only deemed to be so-called *pseudo-anonym* since disclosing such a linkage immediately and irrevocably discloses the anonymization of the entire transactional history.

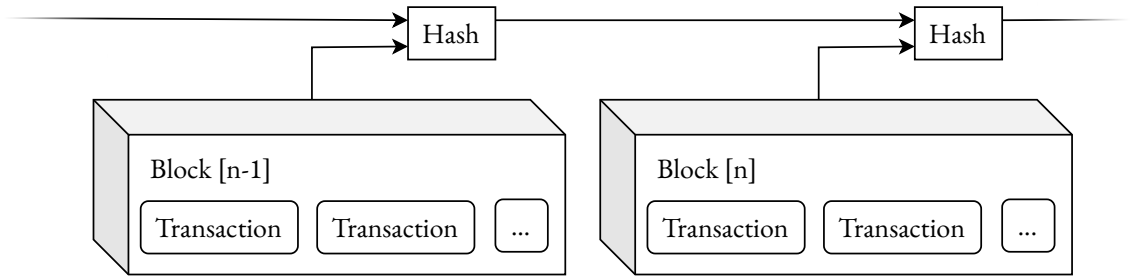

FIGURE 2.2: Bitcoin Blockchain
Packages or blocks of data together with the previous block hash value serve as input for the next block hash value building a chain. Figure inspired by Nakamoto [3, Chapter 3].

2.2.6 Mining

The term *mining* subsumes the process of creating new blocks and propagating them into the network so that they can be attached at the top of the current chain of blocks. Every entity in the Bitcoin network is free to participate in this process if desired. However, it induces specific capabilities to be economically efficient. The Bitcoin protocol, which all the nodes involved follow independently, enforces certain requirements that a new block must meet to be accepted by the network. This process is also known as *finding consensus*.

These requirements include, for instance, a maximum block size of 4 Megabytes and a minimum amount of leading zeros in its block hash value. As previously discussed, a cryptographic hash function is an unpredictable, chaotic one-way function that makes it impossible to use any strategy while hashing for the desired output other than extensive search[8]. Therefore the requirement of leading zeros that a feasible block hash requires is also known as *difficulty target*.

The target is constantly readjusted according to the mean block time $\bar{t}$, i.e. the time passed until a new block was found, of the last 2'016 block epoch. Bitcoin aims for $\bar{t} = T^\star$ whereas $T^\star$ equivalents 10 minutes. An epoch with $\bar{t} < T^\star$ will increase the difficulty target accordingly, resp. decrease when $\bar{t} > T^\star$. This mechanism ensures in the long run that new blocks are found at a constant rate regardless of the invested computation power, also known as *hashing power*. Consequently, these epochs of 2'016 blocks approximately corresponds to $\frac{2016*10}{60*24} \approx 14$ days. It follows that Bitcoin is not only a peer-to-peer money system but also represents a universal clock with an average precision of 10 minutes.

Miners can freely choose which pending transactions they want to include in the block they are currently working on. On average, a single block includes approximately 2'000 transactions[9]. However, there exist blocks that only include a handful or even no transactions at all. Miners also add the missing transaction output that will resolve the earlier mentioned delta between the input and output sides. Doing so, they collect all transaction fees at once. Consequently, miners prefer transactions that reward them with higher deltas or fees, resulting in natural transaction prioritization.

The total transaction fee, however, is not the only incentive to mine. The first transaction of each valid block is the so-called *coinbase transaction* which differs from regular transactions. It exhibits no inputs and only one output spendable by the entity that has mined this block. Hence, coinbase transactions embody the issuance of brand-new Bitcoins which increase the total supply and thus are referred to as *block reward*.

The amount of newly issued Bitcoins in this output-only transaction, however, underlies strict requirements enforced by the consensus-oriented network and is continuously halved as time progresses[10]. Combined with the difficulty target adjustment explained above, Bitcoin is therewith able to control and enforce its supply autonomously.

[8] Also known as *brute-forcing*.

[9] See this Bitcoin Blockchain Explorer for more block statistics.

[10] Block reward is halved every 210'000 blocks $\approx \frac{210000*10}{60*24*365} \approx 4$ years (e.g. BTC 6.25 as time of writing).

Would a miner try to bend these consensus rules to its favor, the invested work in the form of the extensive search after a suitable block hash value would be squandered since no protocol-adhering node would accept this block. A block will also be rejected by the network when it contains invalid transactions. Additionally, the node that issued this cheating attempt will shortly be isolated as the connected nodes will stop communicating with it.

2.2.7 Wallet

Wallets are the primary tool for users to interact with the Bitcoin network. They implement the standards and functions defined in Bitcoin in the form of software applications or dedicated hardware devices. Generally speaking, wallets qualify as such when they exhibit the following two functionalities.

First and foremost, wallets have access to private or public key information which makes them highly critical in terms of security. It allows the wallet to scan a copy of the entire blockchain and search for UTXOs that potentially can be unlocked resp. spent. The summation of all spendable outputs found in the timechain of transactions is conventionally displayed as a *balance* to the user, alongside a transaction history. There are use cases in which wallets only have access to public key information and thus operate in read-only mode, for instance, to observe balances.

Secondly, wallets provide a user-friendly and robust interface to create valid Bitcoin transactions. Robustness is vital since a tiny typo could lead to the irreversible loss of funds. The level of technical abstraction in the transaction generation process varies among the different implementations available nowadays. Thanks to the defined standards in Bitcoin, wallets abstract all the technical details from the user and solely use simplified terminology such as *receive* or *pay*. As in any other high-tech area like smartphones, cars, or medicines, simplifications through abstraction are crucial to encourage adoption. More advanced wallets, however, allow for a user-defined selection of UTXOs in the transaction assembly process and complete fee control.

To conclude, wallets are the primary tool to analyze the public timechain of transactions bundled in blocks as well as to create and propagate new transactions that consume spendable UTXOs as input and declare new UTXOs. Since wallets usually have access to private key information, conscientious management is crucial, which is why this topic is the focus of this work.

2.2.8 Bitcoin Improvement Proposal

Bitcoin's further development is managed by its community which is open to everyone. However, certain formalities for proposing changes should be respected as the unification helps to maintain clarity and consistency for all involved parties.

Therefore the concept of the *Bitcoin Improvement Proposal (BIP)* has been established[11]. It guides authors to a throughout and details description of their proposed change to improve the Bitcoin protocol.

Basically, all discussed Bitcoin functionalities in the following chapters directly or indirectly relate to the detailed specifications provided in the form of one or several BIP(s). Where appropriate, the relevant BIPs will be referenced in this work to enable the readers to investigate these technical topics further. It must be said, however, that BIPs usually require a profound knowledge of other technological fields and/or preceding improvement proposals. Consequently, it can be challenging to fully comprehend a BIP as it generally targets a different readership than this work.

[11] See the complete BIP list hosted on in a designated GitHub repository.

3 Beginner

This chapter discusses two common ways new users typically start interacting with Bitcoin. The first relates to the different forms of online accounts. The second represents a basic form of wallet. Both are explained in detail with the main advantages and disadvantages.

3.1 Online Accounts

Individuals who start cultivating an interest in Bitcoin will probably begin with internet research. Although every aspect of Bitcoin is freely available on the internet, it can be challenging to identify authentic resources. Search results of cryptocurrency exchange platforms are likely to appear in the top listing as they are promoted by revenue-oriented companies[1] paying for advertising. They provide access to numerous different cryptocurrencies or other trading products such as derivatives[2]. Ultimately, every transaction performed on their platform will marginally contribute to their revenue and is usually more expensive. Though these platforms offer various digital assets, this work focuses exclusively on Bitcoin. Likewise, the wide range of companies providing such services is simply referred to as platforms.

These platforms have in common that they all require a user account in order to start interacting with Bitcoin. Nowadays, such account creation processes most often involve any form of proof of identification as they underlay the so-called *Know Your Customer (KYC)* guidelines. Depending on the physical location of the platform provider and the applicable jurisdiction, the KYC measures may vary in their seriousness. These guidelines aim to diminish the risk of money laundering or other criminal financial activities. Proof of a customer's identity can be provided, for example, by presenting a copy of an official identification document, e.g. a passport or driver's license. The linking of a bank account to the user account of the platform can implicitly serve as KYC-compliant identification since, in such cases, the platform relies on the prior identity verification of these banks.

3.1.1 Tradeoff

Besides a dominant presence on search results pages and social media, customers of such platforms generally profit from the competitive user experience on both web and mobile applications. Trading and accumulating Bitcoin appeals foolproof. However, abstracting the complexity of Bitcoin from the user entails inevitable tradeoffs that may seem obscure to beginners in the field.

[1] For instance Coinbase, Crypto.com, Kraken, Bitfinex, or Binance.
[2] Category of trading products whose values are derived from other products' price development.

Custodianship

The highest cost is that platforms generally possess the custody of a user's Bitcoins. Therefore, these platforms are also referred to as *custodians*. It implies that they act as a protective guard between the individual user or customer and its associated UTXOs. This proxy situation is a critical difference to understand compared to non-custodial wallets. Balances shown to logged-in users must not necessarily exist as a set of UTXOs in the underlying Bitcoin blockchain. Sometimes users do not even receive access to private or public key information that would allow them to verify their transactions independently and directly. These online accounts are therefore also called *hot wallets*, which should indicate the high risk of loss of funds.

History has repeatedly shown that the necessary trust placed in platforms is prone to abuse [30, 31]. Forms of such abuse include, for example, poor security measurements that allow for successful hacking attacks or so-called *exit scams*. Latter describes the phenomena where a platform's web application suddenly disappears, in most cases together with its founders.

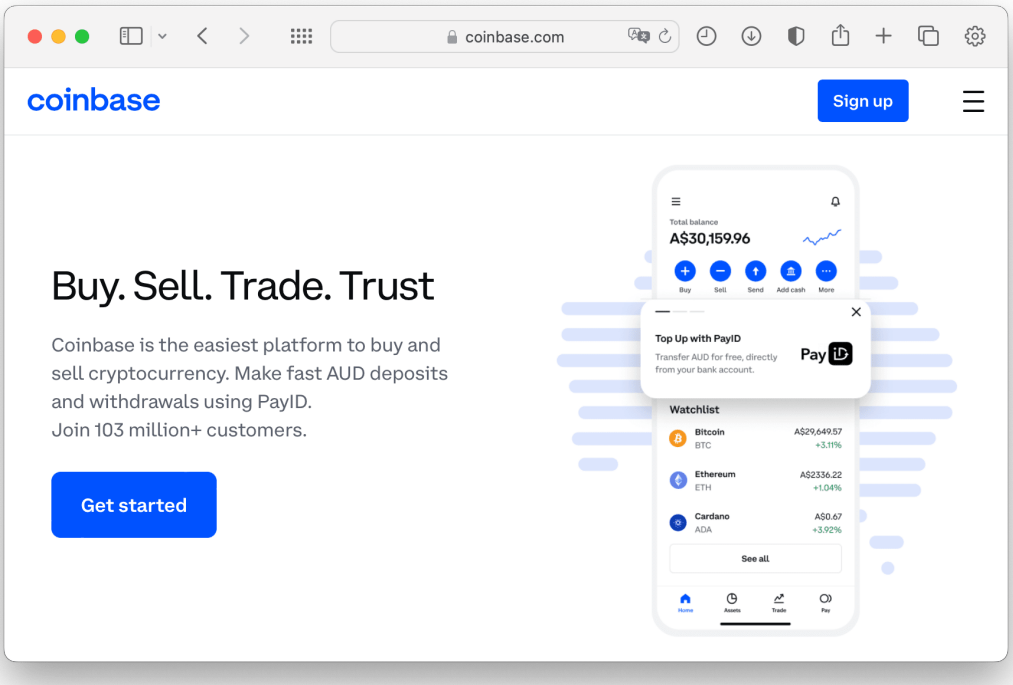

FIGURE 3.1: Homepage of Coinbase
Shows the homepage of Coinbase, a popular custodial service provider for digital asset management. As the headline suggests, such online account wallets require trust due to their custodial characteristic. Image taken on 30. October 2022.

Privacy

Another cost concerns diminished privacy. Equally to all Bitcoins underlying the platform's custody, so do all interactions with Bitcoin by the user. Often there is no other option than interacting with the platform's web or mobile application. The receipt of traffic in the application by identified and logged-in users paves the way for collecting significant amounts of user data. This is not only technically easy to implement but also economically rewarding in times when data is considered the new gold.

Besides user statistics, platforms also have complete control over a user's Bitcoin transactions. It is not self-evident that exchanges consistently act neutral and fair to everyone. Blocking users, freezing funds, and censoring transactions are examples of how platforms can infiltrate someone's freedom of expression, as happened in connection with the 2020 Twitter account hijacking attack[3] [32, 33].

Attractive Target

Another cost associated with these platforms is the increased risk of cyber attacks. The larger the amount of capital managed by a platform for its users, the more attractive it becomes for cyber criminals seeking to exploit potential vulnerabilities for financial gain.

In the worst case, attackers receive access to private key information, which consequently allows them the unlocking UTXOs. A valid transaction would then immediately transfer existing UTXOs to the attacker's public key address. Once such a perfectly legitimate-looking transaction is attached to the blockchain, the affected UTXOs will be irrevocably lost for the platform, respectively its users. Attackers might also profit indirectly through stolen user data. The enforced KYC process has the side effect that user accounts usually include highly sensitive and thus valuable information.

3.1.2 Reasons for Justified Usage

Despite the significant disadvantages and risks that emerge from hot wallets, certain use cases still justify their existence. The following paragraphs list the most noticeable ones.

Customer Support

The first reason that justifies custodial services lies in the nature of Bitcoin itself, namely its complexity. Managing funds independently and securely using non-custodian solutions requires a certain level of understanding in this domain. It would be illusory to think that anyone interested in Bitcoin is willing to invest the time and effort required to acquire a sufficient understanding. Hot wallets still allow them to participate in this decentralized payment system. Additionally, custodial services usually provide some sort of customer support services. This can boost the confidence of novice users in securing their funds.

[3] See 2020 Twitter account hijacking attack for details.

Better User Experience

Other reasons to choose a custodial over a sovereign wallet may lie in the available applications. Profit-oriented companies usually employ designated workforces that optimize how users interact with their platforms through a *Graphical User Interface (GUI)*. Therefore, such applications are generally ahead of open-source solutions regarding usability and design. This automatically makes them more appealing to inexperienced users.

Additional Services

Lastly, additional services or products that enhance the Bitcoin interaction can also persuade users to decide against a sovereign wallet solution. Such services may include management of other assets, financial consulting, expense analysis, or taxation reports. Another everyday use case involves deposits or withdrawals from resp. into the more conventional currencies. However, there exist technologies that aim to fulfill this use case also in a peer-to-peer setting[4].

3.1.3 Take-Aways

The benefits of abstracted complexity through a platform application provided by a for-profit company invoke the cost of custody, loss of privacy, and need to trust that the platforms protect their application against cyber attacks. These costs are often seen as too high for intermediate or advanced Bitcoin users as they possess the know-how to autonomously interact with Bitcoin in its intended manner, which is entirely trustless.

Nonetheless, legitimate use cases exist that still justify using custodial services such as other trading products or services. It is also worth mentioning that platforms can support their user base and, last but not least, represent legal entities that can legally be prosecuted if circumstances require.

3.2 Paper Wallets

The previous section elaborated on the pros and cons of relying on custodians through hot wallets. Doing so, one neglects the most valuable feature of Bitcoin which is the possibility to not rely on anyone. In order to profit from this feature, this section discusses the procedure to create and subsequently interact with a simple but sovereign wallet.

As learned in the Prerequisites, the only thing required to interact with Bitcoin is simply a huge, randomly chosen number that acts as a private key. Using the private key, one derives a public key and, eventually, the associated Bitcoin address. Latter is used for searching the entire time chain of chronically bundled transactions to determine one's UTXO set, i.e. Bitcoins to be spent or unlocked. The term of *creating* a wallet subsumes this process. After successful completion, it is advisable to *backup* the wallet, i.e. saving

[4] For example Bisq, Hodl Hodl, or Peach.

the private key at a secure, preferably offline location. A backup allows for *restoring* the wallet, i.e. reading the private key information from the backup for any future Bitcoin interaction, for instance, on a new computer device.

3.2.1 Generation Methods

Since everything is simply information, it was common practice during the early days of Bitcoin to print the single private key alongside its public key on paper which coined the term of *paper wallets*.

An example of how such a wallet can look is shown in Figure 3.2. Usually, the design is optimized to print the image on physical paper, which resembles a conventional banknote. It is also common practice to QR encode both key parts for more convenient interoperability.

The term paper wallet describes the situation in which only a single key pair is known to the user, also known as *single-address wallets*. This work relies on this definition and uses the two terms interchangeably. However, other definitions consider any keying material printed on physical paper as a paper wallet, regardless of its quantity. The following paragraphs will explain three methods to generate such wallets.

Analogue

The above elucidated wallet generation procedure could theoretically be performed manually using paper, pen, and dices [34]. Likewise, one could manually scan the entire blockchain for spendable UTXOs in order to become acquainted with one's balance.

FIGURE 3.2: Printable Paper Wallet
Example of a printable paper wallet whose design is inspired by physical banknotes. The QR code on the left-hand side encodes the public key address, which is used to receive or load funds and verify the total balance. The QR code on the right-hand side encodes the private key required to create the unlocking script that allows spending the funds.

A significant advantage of such an analogue key generation process, if conducted in an appropriate environment, lies in the vanishing small probability that eavesdropping attackers compromise the keying material since no computer-aided devices are included that potentially could be infected by spyware[5]. However, the disadvantage of an analogue procedure is that it is significantly time-consuming and circuitous. It will likely cost an average person several hours tossing dice and performing written computations to obtain sufficient entropy.

Software-Aided

Another key generation method introduces the help of computers and is thus denoted as *software-aided* in this work. Most users prefer computer tools to generate the key pairs automatically, as it introduces two major advantages over the analogue technique.

The first advantage concerns the speed, as keying material can be created within milliseconds instead of hours. The second advantage reveals in the ease of implementing a key generator or utilizing one of the many open-source solutions available. A critical aspect of such tools hides in the quality of the utilized *Random Number Generator (RNG)*. Keying material that emerges from poor randomness facilitates brute-force attacks through increased predictability and should, by all means, be avoided.

Online Services

The last category of key generation methods concerns online services. Although declining in popularity, there still exist webpages that offer their users to generate highly secure private keys. Sometimes the users are demanded to contribute to the entropy, for example, by random typing or cursor movements. The danger lies in the fact that one must trust that the website provider does not store a copy of the issued private key to steal the accumulated UTXOs at some point in the future.

Other threats that apply when relying on online paper wallet services include malware-infected browser extensions, manipulated random number generators, or eavesdropping entities during server-client communication. Though the usability of such services seems appealing to Bitcoin beginners, one should desist them by any means.

Take-Aways

Paper wallets describe the simplest form of Bitcoin wallets consisting of a single private/public key pair. Although it is possible to generate such a key pair completely analogue, relying on transparent computer tools is recommended. Transparency is vital in order to verify the quality of underlying RNG that are essentially responsible for maximizing entropy. Keying material should never be generated using online services as the risk of receiving a compromised private key is high.

[5] Software that secretly gathers information on infected devices.

3.2.2 Reasons for Discouragement

Though the former popularity of paper wallets, it is nowadays no longer promoted since the disadvantages outweigh the sole advantage of its simplicity. The reasons for this are rooted in the fact that paper wallets are solely associated with one single-address, which promotes the reuse of this address for multiple transactions. Address reuse, however, is a source of critical drawbacks discussed in the following paragraphs.

Privacy

The first issue of paper wallets concerns the owner's privacy. As previously stated, the public availability of the entire transaction history embodied in the blockchain downgrades the quality of anonymity to pseudo-anonymity. Transacting with other entities forces a paper wallet user to disclose the only available public key address to these entities. This immediately links the entire transaction history connected with this address to the user's real-world identity.

For example, when Alice refunds the price of previously paid coffee to Bob, she will ask Bob to share his Bitcoin address P with her so that she can issue the appropriate transaction. Consequently, Alice knows that Bob holds the private key from which P was initially derived. This insight allows her to associate every past transaction and the current balance assigned to P with Bob.

Presenting a Bitcoin address to receive funds inevitably acts as strong evidence[6] that the presenting entity possesses the respective private keys to it as the entity has an interest in spending the received UTXOs at some point in future. The de-anonymization of Bob's activities from the example above continues with every new entity he wants to transact. Any entity that has ever transacted with Bob is henceforth able to track all of his past and future transactions closely. For privacy advocates, this represents a significant disadvantage of single-address wallets.

Censorship Resistance

Linking a real-world identity with a Bitcoin address destroys its pseudo-anonym characteristic. The resulting privacy infringement, however, is not the only danger connected to this. It also opens the doors for actions to censor transactions that involve such identified addresses. Authorities such as governments maintain public lists[7] for which they have declared any form of interaction illegal.

Censorship on Bitcoin transactions cannot be enforced as directly as in conventional payment systems, where authorities can immediately freeze accounts or no longer process certain transactions. Creating a transaction and autonomously propagating it to the decentralized network of independent Bitcoin nodes is comparable with the practice of

[6] As will be explained later, a public key presenting entity must not necessarily be the (sole) owner of the respective private key.

[7] See an example of such a list from the US Department of the Treasury. «XBT» prefixes Bitcoin addresses.

free speech. WikiLeaks' donation story exemplified this [35]. However, censorship efforts regarding banned addresses may be indirectly enforced through regulations to which institutions professionally associated with bitcoin must adhere.

Security

Single-address wallets, moreover, introduce a significant disadvantage that directly affects the security of the private keys and can therefore be considered the most delicate. As stated in the Prerequisites, consuming UTXOs, i.e. using them as input for a new transaction output, typically requires a suitable digital signature that satisfies the underlying locking script condition(s). Doing so, however, reveals further details of the underlying signature scheme that has so far remained unknown to the public. It is essential to understand that the disclosure of this new insight occurs only at the moment of spending transaction outputs. This does not affect the reuse of an address to receive funds repeatedly.

In theory, a form of attack is based on this new information and technically facilitates recovering the private key using a brute-force approach. To the date of this work, such an undertaking costs far more time than it costs for the transaction to be settled. However, having access to multiple different scripts that unlock transaction outputs of the same public key address facilitates these brute-force attacks even more. Hence, one should, by all means, avoid address reuse.

Further threats exist that are associated with the signing process, however, they are only partially exploitable by reusing addresses. Instead, it concerns the applied signing function itself, more precisely, its implementation. The signature's quality relies, among others, on the RNG used for the creation process. As previously stated, weak RNG produces less entropy which eventually facilitates guessing attacks.

Take-Aways

Address reuse destroys a Bitcoin user's privacy and allows authorities to censor interactions with that address. Although such censorship cannot be enforced directly in Bitcoin, it can affect users indirectly through regulations that other institutions must adhere to. The most extreme risk, however, is that the reuse of addresses and the information published with them facilitate specific attacks that potentially result in the loss of one's private key to others. Therefore, using paper or single-address wallets should generally be discouraged.

4 Intermediate

By now, two common forms of Bitcoin interaction possibilities through wallets have been elaborated. These two, namely custodial services and paper wallets, may satisfy the needs of new users in this field on a beginner level. However, the presented disadvantages can distract users with a higher standard. The content of this chapter accounts for such intermediate users.

The price of the simplicity gained through online services or single-address wallets is usually paid with diminished privacy, autonomy, and, eventually, security. Luckily, alternative ways of Bitcoin interaction serve the purpose of restoring these features while keeping the user experience on a comparable high level. The following two sections will discuss such alternatives more concretely.

4.1 Software Wallets

The only advantage of a single-address wallet, namely its simplicity, is outweighed by the various disadvantages that evolve from using it. A natural solution to circumvent address reuse would be gradually creating new pairs of keys whenever a preceding pair was involved in a transaction. More specifically, this could happen before a spending transaction is published so that the resulting change can be transferred to a new Bitcoin address that has never been involved in transactions before. Likewise, UTXOs from many different addresses could be periodically merged to a new one using a consolidation transaction, i.e. a transaction with many inputs and only a single or few outputs. The rightmost transaction in Figure 2.1 visualizes such a merging transaction, for example.

Successive key pair creation thus represents a feasible solution to the problems induced by address reuse. However, applying such a strategy no longer preserves the simplicity known from the single-address approach, as one must properly manage several key pairs from now on.

Managing multiple key pairs does not only imply a more comprehensive backup procedure but also increases the complexity to create transactions since UTXOs from different independent sources must be considered. Since Bitcoin is a purely digital concept, one can write computer programs that support these processes using control mechanisms and automatization. This section will be dedicated to these kinds of software.

Before the initially proposed strategy of successive key pair creation is explained in detail, the essential characteristics of such computer programs are defined.

4.1.1 Definition

Generally, one can define everything that serves the purpose of storing a private key as a wallet. The previously discussed paper wallets originally accomplished this by printing the sensitive information on physical paper. Another and more popular form of wallets store the keying material within computer applications and are thus referred to a software wallets. More precisely, this means that private keys reside on a computing device such as a laptop or smartphone in such a form that a given software can access and work with it using a GUI. Figure 4.1 exemplifies such a GUI to create and broadcast a transaction.

So far, the association of software with wallets has already been stated in the previous chapter. However, in this previous context, it was only meant to be used in the wallet generation procedure, which must be differentiated. Software wallets on the other hand qualify as more sophisticated since they usually provide a range of functionalities that facilitates the overall interaction with Bitcoin to great extent.

Many different implementations of software wallets evolved over the past years[1]. Although the technical details may differ, they all implement certain basic functionalities. The most important ones are outlined below.

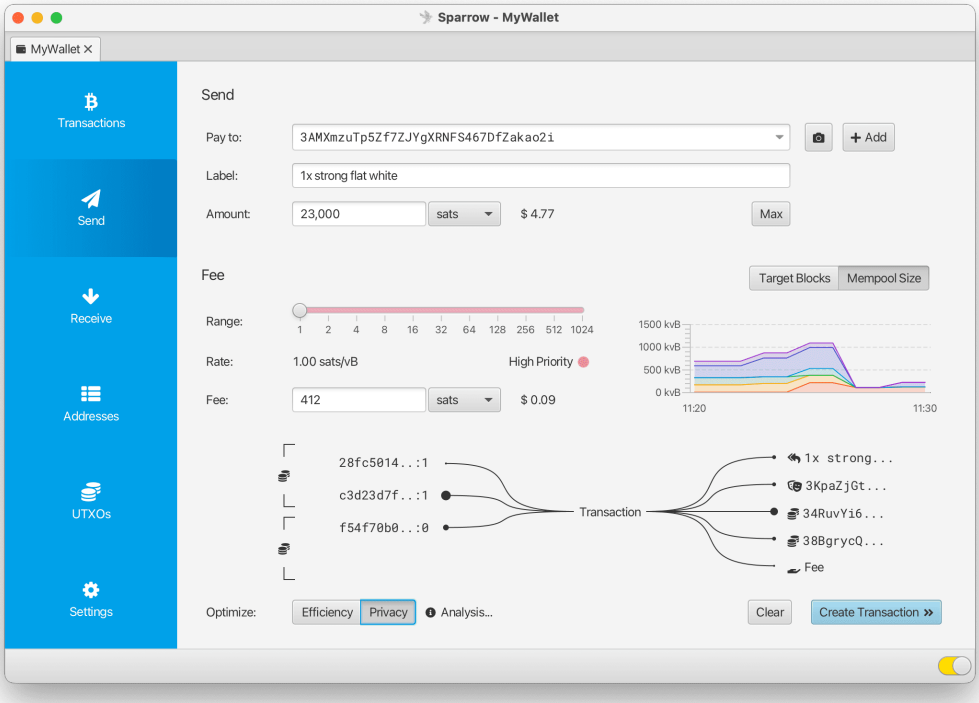

FIGURE 4.1: Sparrow Wallet Application View
Shows the transaction creation view of the software wallet implementation called Sparrow. The basic information required is the beneficiary address «`Pay to`» and the «`Amount`» to be sent. The «`Label`» allows for a personal note for better recall purposes. Additionally, a detailed fee control and input-output visualization is offered.

[1] Software wallet comparisons by bitcoin.org, Veriphi, or WalletMatrix serve as comprehensive resources for recommendations as well as selection assistants.

Key Storage

First and foremost, software wallets store keying material, which serves as a basis for all subsequent functionalities. The measurements to protect entered private keys may vary. Some implementations, for example, store an encrypted version of the key on a storage disc, while others simply write it as plain text into a file.

Another important aspect of this key storage functionality is how the keys can be imported or exported. Popular implementations provide specific workflows for such key transportation, which may even allow for additional protection through intermediate encryption.

As already mentioned, software wallets can generally be used in read-only modus as well by solely providing the public key information. Although this eliminates the risk of private key loss, the user's privacy might still be compromised since wallet usage data can be linked to the underlying public key(s).

UTXO Management

A software wallet's next central task is managing a current set of relevant UTXOs. In other words, it represents the wallet user's current Bitcoin balance. The involvement of the keying material known by the software wallet makes a transaction relevant in this sense. More specifically, it will initially scan the entire blockchain and every new block for transactions involving the hash value of the public key. The chronicle of the receiving and spending units serves the application as a basis for the balance derivation.

Additionally, the wallet application also observes the set of unconfirmed transactions, i.e. transactions which are inherently valid but not yet included in a block to be attached to the blockchain. This pool of valid but unconfirmed transactions is also known as *mempool*.

Transaction Creation

Last but not least, software wallets implement standardized techniques to assemble valid Bitcoin transactions and populate them to the network of nodes. This is also the functionality that can vary the most among the many different implementations available nowadays, usually recognizable in the extent of the setting options and features. An example of how this could look is shown in Figure 4.1.

For example, the minimal user input required to create a valid transaction would be the recipient's public key address and the amount of Bitcoins to be transferred. However, simplifying the transaction creation process abstracts the richness of transaction features by hiding them from the user.

One such feature is, for example, the choice of transaction fee. Complete control over the fee payable to the mining entity that chooses to include this transaction in a feasible next

block allows for custom prioritization[2]. At a certain interval, higher fees can be said to increase the probability of a transaction being included in the subsequent n blocks where n is minimized. Low fees decrease this probability, respectively increase n.

Another important aspect of the transaction creation process is the change addresses. As in the real-world, paying in cash most often results in receiving change. It lies in the responsibility of the wallet implementation that the change in the form of another UTXO remains spendable using the known keying material. Imagine the worst case scenario where an improperly implemented software wallet consumes an UTXO of BTC 5.0 to pay for a meal worth BTC 0.001 and, neglecting fee, creates a mistakenly change UTXO of BTC 4.999. This change could no longer be spendable using the known keying material and would be lost forever. Luckily, standardized procedures for change address derivation exist, which will be discussed in further detail at a later stage.

Take-Aways

Software wallets represent the main interaction medium to Bitcoin nowadays. Thanks to detailed documented standards, everyone can implement the basic features required for this interaction. The various wallet implementations available to date share a lot of standard features. Such basic features include import and export mechanisms of private and/or public key information and the observation of historic and new transactions involving the given keying material.

However, software wallets differentiate more significantly in the level of abstracting more advanced features in the transaction creation process. Fee control and change treatment are examples. Thus it is imperative to investigate and test the functionalities of software wallets before using them in production.

4.1.2 Wallet Derivation

With the definition and role of software wallet known, the successive key pair creation strategy can be addressed. As previously mentioned is the most trivial approach to escape the various disadvantages spawned by address reuse to successively create new key pairs to be used. The following paragraphs describe three different techniques for implementing such a consecutive key generation approach.

Non-Deterministic

In the early days of Bitcoin, there existed software wallets that implemented this strategy by simply repeating the random key pair generation process and keeping a record of it. This procedure is considered to be *non-deterministic* as the same input (e.g. the first private key) does not result in the same output (e.g. set of consecutively generated key pairs).

[2] Johoes's Bitcoin Mempool Statistics is a recommended source for current fee market insights.

Generating sequences of private keys using randomness can technically not be declared as a wallet derivation technique, as there exists no constant from which the same sequence can be derived repetitively. An arbitrary collection of private keys K_i is essentially the result, as modeled by Figure 4.2 in the wallet image A. This promotes the importance of knowing the individual keys as such. Without a backup of all private keys ever created by the application, a full wallet recovery would be impossible on a new computer device. In other words, non-deterministic wallets lack replicability and should thus be avoided.

Deterministic

A second and much better approach represents a *deterministic* key creation procedure where a given input repetitively results in the same chronic of consecutive outputs. The input in this context is commonly referred to as *seed*.

Using a seed S and a deterministic key generation function f, one can derive the first private key $K_1 = f(S)$. The next private key could then be derived on behalf of K_1 such that $K_2 = f(K_1)$. The image B in Figure 4.2 visualizes this. Another option would be a function that not only takes S as an input but also an advancing counter i so that $K_1 = f(S, i = 1)$ and $K_2 = f(S, i = 2)$. Both key derivation methods will repetitively result in the same chain of keys as they are rooted in the given seed.

The advantage of such a deterministic approach compared to the non-deterministic one is that only the seed S and the deviation function f must be known in order to recover any funds related to any K_n. However, this advantage introduces critical disadvantages concerning the ease of creating a backup.

While knowing S is relatively straightforward, receiving full transparency on the inner working of f can be challenging, depending on which software wallet implementation is used. In the worst case, a user relies on a closed-source wallet application and thus not even has the chance to understand f. The success of recovery attempts using different wallets is not guaranteed as the implementation of f may differ. It is a reminder of the importance of accessible software source code for Bitcoin applications to promote complete transparency.

Hierarchical Deterministic

Deterministically deriving any amount of key pairs from a piece of single secret information in the form of a seed has become common practice for Bitcoin users nowadays. Bitcoin provides a standard deviation technique known as *Hierarchical-Deterministic (HD)* wallets[3]. Here the deviation paths are no longer linear but resemble the branching paths of a tree, i.e. hierarchical.

Wallet image C in Figure 4.2 visualizes this. The master key pair K^M represents the first node that is derived from the initial seed. A node in this context can be seen as a bifurcation

[3] See BIP32 for details.

point in the hierarchical derivation structure. From here n different so-called *wallet accounts* are derived, denoted as K_n^M. Every account splits into an external and internal so-called *wallet chain*. Public keys of the external part $K_{n/0}^M$ are meant to be communicated to other parties, while the internal wallet chain $K_{n/1}^M$ is exclusively used for change addresses. Both chains serve as a basis to derive i various Bitcoin addresses $K_{n/\{0\|1\}/i}^M$. The latter can also be seen as the leaves of this HD wallet tree. Meanwhile, more conventional deviation paths exist[4] that, for example, have additional levels for indicating a purpose.

While sharing keying material of ordinary deterministic wallets is an all-or-nothing decision due to its linear characteristic, HD wallets allow for partial sharing as multiple chains of derivation paths exist. Using the metaphor of a tree, following the trunk will lead to all leaves on all branches, while a single branch only leads to leaves on that specific branch. The Bitcoin standard promotes standard derivation paths, which are usually implemented as default in software wallet applications.

Derivation paths are particularly useful as only public key information can be shared at a custom level in the tree hierarchy. These so-called *Extended Public Key (xPub)* make it possible to continue the address hierarchy towards lower levels [5]. For example, a company owner could only release the public key information of a specific wallet account to the department responsible for the online shop. This capacitates the department to continue deriving unique addresses for every purchase committed in Bitcoin without ever having access to private key information as well as receiving insights over the complete HD wallet structure.

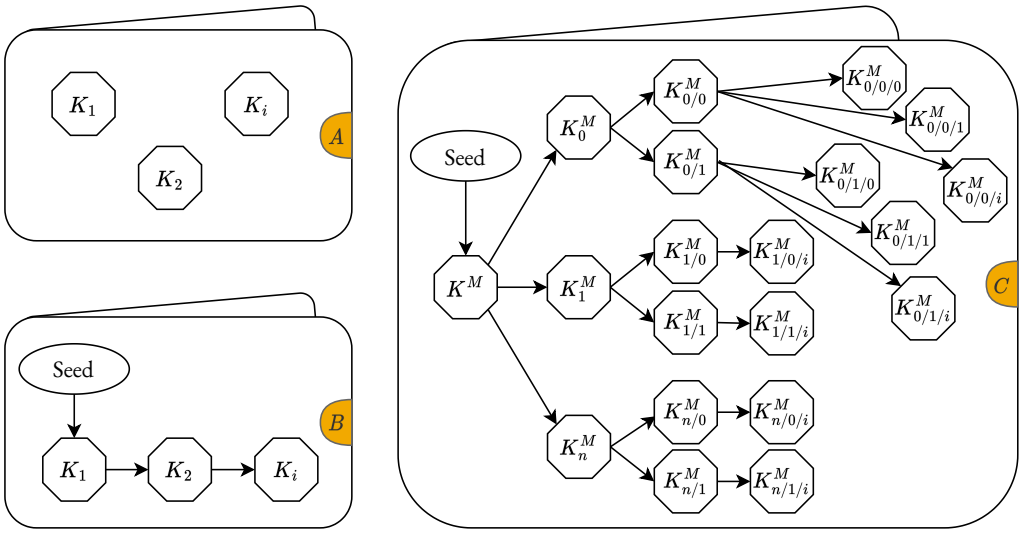

FIGURE 4.2: Wallet Derivation Types
The repetitive random generation of key pairs shown in wallet *A* is technically not deemed a derivation technique and thus discouraged from being used. Wallet *B* allows for a seeded linear key derivation path. The derivation technique modeled in wallet *C* also uses a seed but develops a hierarchical path structure deemed a wallet tree. Figure inspired by Antonopoulos [24, Chapter 5].

[4] See BIP44, BIP49, BIP84, and BIP141 for details.
[5] Extended public keys based on other standardized derivation formats include `yPub` and `zPub`. However, for simplicity, the terminology used in this work is limited to xPub.

It is important to note that the proposed deviation paths and their meaning act as a convention. Prominent software wallets that support HD wallets adhere to this standard by default. This implies that a seed backup alone is enough to recover funds in the entire tree structure, regardless of the used implementation.

Some implementations, however, allow specifying user-defined deviation paths. Doing so makes it crucial to back up the specified paths alongside the seed, as otherwise, a recovery attempt would not scan the correct tree structure for spendable transaction outputs. It also decreases the wallet's compatibility with other software since particular implementations may not account for such a custom path input. Except for purposes of intentional obscurity, there is no justifiable reason for utilizing a custom deviation path in practice which is why it should be avoided.

Seed Phrases

Central part of every deterministic wallet implementation is the seed itself as everything is rooted in it. It became common practice to encode this seed using 12 or more words originating from a specific set of feasible words[6]

Table 4.1 shows two example seeds consisting of 12 resp. 24 words. The order of the words is crucial and aligns with the reading direction, i.e. from left to right and line after line. The feasible words from the dictionary exhibit some crucial features. First, they are unambiguous and easy to spell. Second, every word in this set is uniquely identifiable through its first four characters. Together these features should facilitate the backup process while reducing human errors.

The initial rationale behind this idea was to receive an easy-to-memorize sequence of natural words which coined the term of *mnemonic*. Although seeds represented as word sequence are indeed less challenging to memorize[7] it is still recommended to adhere to a physical backup strategy since the reliability of human brains should generally be doubted. Better names are therefore *seed phrase* or *seed recovery phrase*.

The seed phrase can additionally be extended using a user-defined character sequence of arbitrary lengths originally known as *passphrase*. This is not to be confused with a password that software wallets usually ask for in order to encrypt the keying material on the hosting device[8]. Thus, more lucid terms have been established such as *(seed) extension word*.

Using an extension word for e.g. a 12-word seed phrase will result in a completely different HD wallet. Together they qualify as a two-factor security scheme where the actual seed phrase would be something that is possessed and the extension word something that is known or at least located somewhere else. However, splitting the seed phrase into various

[6] See BIP39 for details.
[7] Exclusively memorizing a seed is also known as *Brainwallet*.
[8] Also not to be confused with the private key encryption password specified in BIP38 that acts as a second security factor.

Words	Seed
12	`advance village electric load round goddess search topic click toe vapor voice`
24	`jeans team toe enroll blanket draft current elbow summer wise faculty powder afraid vital cash life jump remember apology fever bullet buffalo sample sword`

TABLE 4.1: Seed Phrase Examples
Shows two example seed phrases based on the word set proposed in BIP39. The words in this set are unambiguous, easy to spell, and uniquely identifiable by their first four characters.

parts is considered a bad practice as brute-forcing becomes over-proportionally easier once certain words are known to an attacker.

Notably, seed extension words allow wallet implementation to maintain two key hierarchies, one without the extension word and one with. Doing so, the user benefits from the implicit feature of plausible deniability in case of a physical attack[9] This is also referred to as a *two-sided* seed, where one side contains only a fraction of the funds that are on the potentially deniable side.

Take-Aways

Deterministic key pair derivation is a superior form of Bitcoin wallets compared to paper wallets. It eliminates the need for address reuse while keeping the backup complexity equal since all derived keys are rooted in a single seed. Following the proposed standard, such seeds consist of 12 or more unambiguous words from a predefined set of words that can be further extended using user-defined characters.

HD wallets introduce further advantages through wallet accounts that, independently from each other, result in different chains of Bitcoin addresses. These wallet chains or tree branches can atomically be shared to access requiring parties without revealing the entire wallet tree structure. Using the standard deviation path promotes both a simplified backup strategy, as only the seed itself must be secured, and increased compatibility with other software wallet implementations.

[9] Can protect against a 5$ wrench attack, which exemplifies that the best technical security measurements may become useless in the event of a physical robbery involving a wrench as a weapon. The name of this attack may change in the future, as wrench prices are also subject to inflation.

4.2 Hardware Wallets

To this point, the concept of how software is used to facilitate the interaction with Bitcoin should be comprehended. Private key information, whether encrypted or not, is nevertheless exposed to the threat of loss as software wallets are usually installed on computer devices that regularly communicate through the internet and have other software installed. Therefore, the wallet technology discussed in this section describes the approach of utilizing isolated computer devices for handling sensitive keying material and further minimizing the risk of loss. Such designated computer devices that solely serve the fundamental purpose of protecting keying material are known as *hardware wallets*. This section discusses this type in detail.

4.2.1 Definition

Hardware wallets are defined by the sole purpose of their existence, which is to secure the private key information stored on them through their tamper-resistant system architecture. Technically, any computing device that serves this purpose can be referred to as a hardware wallet. However, they are usually devices that are specifically designed for maximum security, accessibility, and portability.

Figure 4.3 shows three famous examples of hardware wallets available nowadays. Although they are from three different providers, they show remarkable resemblance regarding their features. For example, all devices are much smaller than a typical smartphone and possess a display.

First of all, hardware wallets should not be able to communicate with the internet. This aspect alone significantly increases the security of the information stored on such devices. Without internet connection the installation of harmful software such as spyware becomes more difficult, let alone the communication of gained insights back to the attacker.

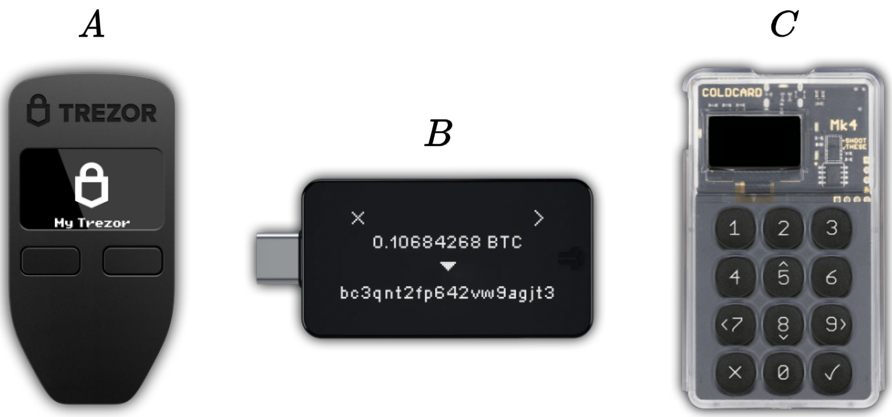

FIGURE 4.3: Three Examples of Hardware Wallets
Device *A* represents the `Trezor Model One`, device *B* the `BitBox02`, and device *C* shows the `COLDCARD`. What all three have in common is the in-built display that is crucial to verify the transaction details before signing.

Additionally, hardware wallets are characterized by their hardened system architecture. Secrets are stored on designated tamper-proof micro-controllers especially designed to make it impossible to extract these secrets in any usable form. The firmware[10] used in hardware wallets are usually open-sourced. Also, they are reduced to their bare minimum to minimize potential attack vectors. Hardware wallets utilize special building techniques and materials that, in case of a physical attack attempt, would either result in the complete destruction of the wallet or unavoidable indications visible to the user.

It is essential to understand that hardware wallets nevertheless rely on software in two ways. First of all, the hardware wallet itself must be able to process the most basic wallet operations. This includes, for instance, key pair generation, address deviation, and transaction assembly using appropriate digital signature schemata. The second way how hardware wallets rely on software is based on absent internet connectivity. One therefore requires an intermediary device that is able to communicate with the detached hardware device and populate relevant information to the Bitcoin network using an internet connection. Most of the prominent software wallet implementations allow for such inter-device communication.

4.2.2 Usage

The Figure 4.4 models the interaction between a secure hardware device D_S and any network-connected computer device D_C acting as an intermediary. The open lock inside the intermediary device models indicates the potentially insecure system environment owed to the active internet connection. An installed software wallet implementation on D_C scans any copy of the Bitcoin blockchain for UTXOs, whereas the relevant Bitcoin addresses were initially queried from D_S. In other words, it is D_S that derives the Bitcoin addresses from the known private key and provides them to D_C.

In order to create a payment, the software wallet on D_C will collect suitable UTXOs and create a so-called *Partially Signed Bitcoin Transaction (PSBT)* using the provided Bitcoin address of the beneficiary(s). Next, this partial transaction will be sent to D_S over the channel α for verification purposes. This includes, for example, verifying the imminent UTXOs to be sent to the recipient address using a dedicated display on D_S. After the verification is confirmed, D_S will finalize the PSBT by creating the unlocking script using the appropriate private key information stored on it. Eventually, the signed transaction is returned to D_C using α, where the software wallet will propagate it to the Bitcoin network using the channel β.

Usually, an *Universal Serial Bus (USB)* or a short-range wireless interface such as Bluethoot® serves as a communication channel between D_S and D_C. End-to-end encryption protocols[11] prevents eavesdropping attacks on α. Some hardware wallets even allow a so-called *air-gapped* communication interface by solely exchanging text files using *Secure Digital (SD)*

[10] Basic software that enables the interaction between software and hardware on a fundamental basis.

[11] For example the *Noise Protocol Framework* which uses DHKE and is seen as secure alternative to the popular *Transport Layer Security (TLS)* protocol.

cards as a channel. Camera-equipped hardware wallets air-gap α by scanning QR-encoded PSBTs and displaying the signed transaction again as QR code on the device's display.

Information shared using channel β usually happens through a conventional HTTP/S connection or by using *The Onion Router (TOR)* for improved privacy. Technically, there exists no need for end-to-end encryption as the channel β transfers no secret information.

4.2.3 Benefits & Pitfalls

Storing keying material on an internet-incapable hardware wallet introduces several benefits over the wallet solutions seen in the previous chapters. A medium that only stores private key information offline is also referred to as *cold storage* resp. cold wallet and thus naturally more secure than hot wallets.

Analogously created paper wallets, for example, fulfill this property just as well and can, therefore, also be regarded as cold storage. However, handwritten private keys on physical paper result in poorer usability than hardware wallets.

This negative correlation between usability and security is a typical pattern in information security as it usually represents a trade-off. Figure 4.5 roughly visualizes this relationship for the wallet solutions discussed so far. It organizes the various solutions based on offline exclusivity (cold) or online access (hot) and the associated user-friendliness. For the reasons already mentioned, certain solutions are more recommended than others.

As the picture suggests, the storage type is not to be understood as a binary decision but rather as a spectrum with a continuous transition. The reason for this lies in the fact that the private information was on a risky device for a short time but is now only stored physically. For example, a paper wallet created with open-source software and then printed out on a home printer is considered «hotter» than a newly initialized hardware wallet, even though both are ultimately entirely offline.

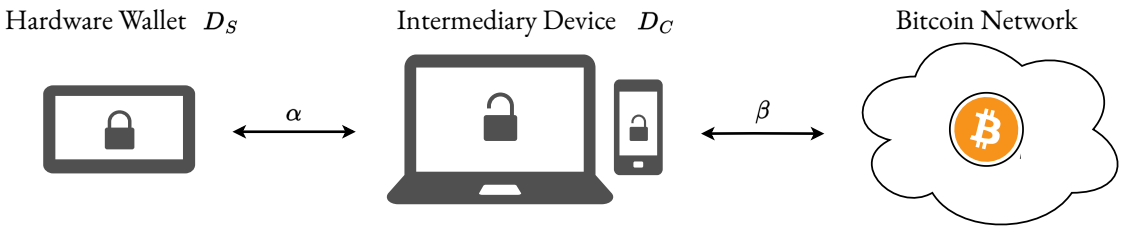

FIGURE 4.4: Hardware Wallet Interaction Model
Private key information is exclusively stored on a secure hardware wallet D_S using a hardened system architecture indicated with the closed lock. An installed software wallet on a network-connected and thus potentially compromised (indicated with the open lock) intermediary device D_C communicates with D_S over a secure end-to-end encrypted channel α. Partial transactions are finalized on D_S before being returned to D_C to eventually be propagated to the Bitcoin network using an insecure channel β.

Security

Hardware wallets are the most secure of all the wallet solutions discussed so far. The entire design, from the firmware to the hardware and the used materials, serves the sole purpose of protecting private key information.

It is crucial to understand, however, that hardware or cold wallets in general are no guarantee for greater security. It remains the user's responsibility to ensure that such wallets are not misused, as doing so may still result in the immediate loss of the private key. As an example, a perfectly analogue created private keys written on physical paper becomes useless if the paper is easy for other people to access or it is not protected in the case of fire. Consequently, using a hardware wallet does not diminish the importance of having a tested backup strategy. They are usually implemented so that, for example, a few wrong pin code repetitions lead to the immediate deletion of all information stored in the wallet.

Although hardware wallets are considered nowadays to be the most secure mean to store private key information, they still do not yield for absolute security. In other words, also hardware wallets can be hacked[12]. For example, an attacker on D_C could tramper a PSBT just before transferring it to D_S. Regardless of whether α represents a wired or air-gap interface, all channel forms would transmit the tampered data equally. Therefore it is crucial always to verify the transaction details shown on the hardware device itself.

Accessibility & Portability

Hardware wallets are also an excellent means to reduce the accessibility to the user's Bitcoin funds. The fact that no transaction can be filed entirely without the presence of

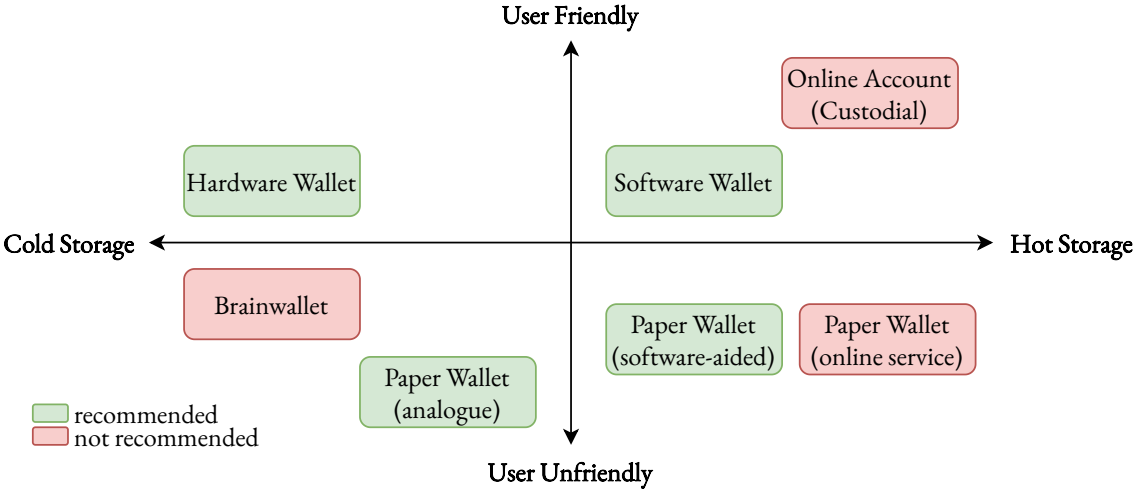

FIGURE 4.5: Wallet's Usability and Security Relationship
Shows the relationship per wallet type between usability and the security given by cold or hot storage. It is important to note that cold storage wallets only provide the potential for higher security through their offline nature. Improper management of cold storage wallets can still lead to losing private keys.

[12] See BitBox02 threat model for example.

the hardware wallet device introduces higher security on a practical level. The device's compact dimensions make them an ideal item to be stored in a safe, for example.

Users that rarely carry hardware devices directly with them reduce the chances of physical attackers further. If the device still needs to be transported, its diminutive dimensions make transportation convenient and inconspicuous[13]. It can also be used on different and possibly compromised intermediary devices. This makes hardware wallets an ideal portable safe, for example, when traveling.

Software Dependence

Despite the explicit name, hardware wallets still host and thus depend on software. For the sake of security, however, the extent of such software applications is usually minimized. The rationale is that the more trivial the application is, the fewer lines of code are required to implement it. Consequently, less code results in less chance of bugs and minimizes the number of potential attack vectors. Nevertheless, hardware wallets exist that aim to support a wide range of different digital assets and thus are forced to install more extensive software on the devices.

Take-away

Hardware wallets are designated tamper-proof devices with the sole purpose of securing the confidentiality of the stored private key information. They are built using hardened firmware and special material design that allow for the detection of physical tamper attempts. Hardware wallets operate entirely detached from any network and are thus considered cold storage. Transactions are partially created using any software wallet on an intermediary network-connected device and transmitted for finalization using a secure communication channel which can be either wired or air-gapped.

Despite the significantly increased security, users must have a working backup strategy. Hardware wallets aim not to be indestructible and have implemented security mechanisms that immediately delete all information stored in the event of a suspected attack. Also, it is crucial always to verify PSBTs on the hardware device itself since they may be tampered by an attackers, regardless of the used channel form.

[13] The COLDCARD wallet shown in Figure 4.3 purposely mimics a pocket calculator.

5 Advanced

To this point, the readership is aware of multiple different techniques regarding the safe management of keying material. It was shown how software wallets consistently derive new key pairs using seeded derivation paths. Also, the concept of hardware wallets was discussed alongside the reasoning for utilizing them. Consequently, the gained insights allow users to interact with Bitcoin on an intermediate level legitimately.

This chapter aims to provide even more advanced insights into the domain of wallets in order to further maximize security and privacy. However, these advanced subjects should be applied only when the user fully comprehends the previous two chapters. Advanced users need to be aware of the increased complexity that these subjects introduce. A complete understanding is, therefore, crucial to avoid the loss of funds due to mismanaged handling of private keys.

5.1 Running own Node

The concept of UTXOs and how software wallet implementations scan the entire timechain of confirmed transactions known as blockchain was discussed in detail in the previous chapters. It follows that the information embodied in these chains of blocks is of great importance and should thus be replicated on many different devices as possible.

Bitcoin's blockchain is not only used to evaluate the total balance of someone's (HD) wallet but also serves as a single source of information regarding the settlement status of a transaction. It is common practice to denote a transaction T as settled and thus confirmed after it was included in a new block B_T and five subsequent blocks were attached on top of B_T. The number of new blocks, including B_T, are also denoted as *number of confirmations* that T has. The rationale behind this 6-block-rule sources in the negligible probability that a longer chain can emerge starting at the block position $B_T - 1$ and thereby no longer include T. While this would be possible in theory, it is considered infeasible in practice as a single mining entity would need to control more than half of the combined hashing power available in the network[1] in order to establish a longer chain that no longer includes T. In other words, the chance that a chain branch becomes the new longest chain and thereby becomes accepted by the Bitcoin network declines over-proportionally with every new block added to the currently accepted chain.

When a user is interested in the current balance of a specific public key address, the question arises of whom to ask for this information. Bitcoin's blockchain is replicated

[1] Also known as *51% attack*.

on numerous autonomously acting network participants, which essentially is the core aspect of its decentralized characteristics. It is crucial to understand that every interaction with Bitcoin will include communication with a node in the network. As a recapitulation, the definition of such a node is provided in the prerequisite's subsection 2.2.1. Software wallets, for example, usually integrate a list of known Bitcoin nodes that are used to place such balance determination requests of end users. This, however, requires the user resp. the utilized software wallet to trust these consulted nodes, which is not a desirable situation.

In order to eliminate the dependency on other nodes and, thereby, the need to trust, one must set up a node by oneself and then use this node exclusively for any wallet communication. More concretely, this implies that the user will download and install the `Bitcoin Core` application[2] on a self-controlled computer device alongside its copy of the entire blockchain, i.e. a *full* node. From then on, the user can route every communication involving any form of Bitcoin interaction through this self-controlled node.

The rest of this section will elaborate on the requirements for such an undertaking as well as discuss the added benefits that emerge from a self-hosted Bitcoin full node. Doing so, the term *self-hosted* node is used interchangeably with *domestic* node while any other node in the network is denoted as *foreign* node.

5.1.1 Requirements

Keeping the setup requirements of a domestic node relatively low is crucial. The reason for this lies in the fact that the low-threshold requirements increase the feasibility of operating a domestic node. This, in turn, contributes to the quality of the decentralization of Bitcoin, which is desirable.

Hardware

The hardware requirements to run a Bitcoin node are trivial and generally affordable for a low three-digit price tag. Bitcoin Core is optimized for computational efficiency and full memory control using the programming language `C++`. Consequently, the hardware requirements are kept to a minimum.

A common approach is to use *Single-Board Computers (SBC)*, i.e. full functional computer devices using a single circuit board[3] to host the Bitcoin Core application. However, any computing device that allows custom software installations and is network-capable is suitable. The popularity of SBCs for this use case reasons by the small dimensions, the absence of overhead installations, and its low energy consumption. The recommended minimum amount of *Random Access Memory (RAM)* storage, for example, is 4 gigabytes which is denoted as sufficient.

[2] Or any other software that is able to communicate to other Bitcoin nodes while respecting its protocol.
[3] The devices produced by the Raspberry Pi Foundation are examples of SBCs.

In addition to the circuit board, on also requires an external storage medium to host a copy of the entire blockchain. As of the time of writing, this replicate requires approximately 400 gigabytes of storage space which will slowly but constantly increase as new blocks are attached. However, Bitcoin's restrictive block size constraint aims to minimize this storage growth rate.

At this point, it is worth mentioning that the use of Bitcoin is legally restricted or prohibited in certain areas on this planet[4]. Hosting computer hardware with the sole purpose of running a full node may expose the host to a higher risk of being prosecuted.

Setup

Once the hardware components are organized and ready to use, the software setup can start. It is crucial to download and compile the required source code individually instead of relying on pre-compiled software packages. The reason for this is that otherwise, the authenticity of the source code cannot be verified using the appropriate digital signatures of its creators.

After successful setup[5] Bitcoin Core will start downloading its copy of the blockchain from foreign nodes. Once this download is completed, it will verify every single transaction in every block, starting with the first block in the chain, also known as *genesis block*. Depending on the underlying hardware specifications, this verification procedure can last between a few hours and a few weeks.

Once this initial verification is concluded, the full node is ready to interact. Broadcasted information, such as newly found blocks or newly issued transactions, will again be verified.

Environment

In order to maximize the additional advantages of a domestic node, specific requirements apply to the physical environment in which it is located. The hosting device should be safe from any unauthorized access as otherwise the integrity of the domestic node could be compromised. Ideally, the selected location is additionally protected against water and fire damage while providing enough ventilation.

Furthermore, domestic nodes should have as little downtime, i.e. time in which they do not operate as intended, as possible. The absence of power or an inactive connection to the Bitcoin network may cause downtime. Latter represents a more delicate issue since wallet communication may still be possible but the information received from the full node is not guaranteed to be the latest.

[4] See legality per countries and territories for details.
[5] Resources for guided installation are provided by RaspiBolt or via the Umbrel operating system.

Take-Aways

Self-hosting a Bitcoin node can be accomplished using trivial computer devices and eliminates the need to trust the information received from foreign nodes. The deployed software should be verified and compiled on the device itself to ensure its authenticity. Ideally, one installs the node in a location that is protected against unauthorized access as well as water and fire damages.

5.1.2 Added Benefits

Users who successfully set up and run a domestic node can benefit from various benefits. However, it is important to note that a node alone only marginally accounts for these advantages. The existence of a full node alone can only be regarded as an advantage in the sense of the improved decentralization of the network. Personal and directly usable advantages only arise when all the wallet communication is routed over the domestic node. Popular software wallet implementations usually account for the use case of connecting to a distinctive node by providing its internet protocol address and port number.

Independence & Trustless

The first and most important benefit that arises from domestic node usage is the increased independence received by terminating any trust-requiring wallet communication to foreign nodes. Consequently, the information exchanged between each wallet application and the ground truth contained in the blockchain is less likely to be tampered with by any third parties. This increases the trustworthiness of, for example, balance information drastically since the user is assured that the calculated total of all UTXOs was produced by a node whose systems have been verified and set up by the user. Of course, the independence gained through a domestic node can only evolve if the domestic node operates as indented in a secured environment.

Furthermore, domestic nodes allow for perfect control regarding system updates. In Bitcoin, this is especially important as the decision on what version or fork[6] is deployed also infers what features are supported. In other words, users that are not satisfied with a specific BIP, for example, can freely choose not to upgrade their domestic node to any version that implements this BIP.

Increased Privacy

A second benefit that arises through the usage of domestic nodes concerns privacy. When wallets communicate with foreign nodes, one must be aware that the node operator can monitor every request. Over time, the operator could draw conclusions about possible credits, since, for example, the same public key address is repeatedly queried from the same internet protocol address.

[6] A continuing software version that was forked from the original one.

The observation of transmitted transactions to foreign nodes is even more damaging to privacy. Payment insights, including metadata, represent highly sensitive information if it can be linked to an individual. This risk of de-anonymization is reduced by communicating a transaction from the wallet exclusively to the home node. If the transaction is then redistributed to foreign nodes, it becomes less distinguishable from other unconfirmed transactions.

The same applies to repetitive address or transaction lookups, such as checking its confirmation status. One should be aware that querying the history of UTXOs of any Bitcoin address using publicly available blockchain explorer services[7] can always leave certain marks on the providers. This is completely avoided when using the domestic node for blockchain explorations.

Take-Aways

Running a self-hosted Bitcoin full node introduces valuable benefits concerning the users' independence and privacy. All sorts of wallet communication should be routed through a domestic node. The wallet user is less likely to receive tampered information, for example, on balances or transaction confirmation status. Additionally, the usage of domestic nodes protects privacy since no foreign node providers can track the communication and try to link it to individuals anymore.

5.2 Multi-Signature Wallets

Simple transactions such as P2PKH or P2WPKH generate a transaction output that can be spent by providing a suitable digital signature generated using one private key. This implies, however, that whoever knows the appropriate private key or seed can unlock the underlying UTXO set. Given the typical scenario that a private key is controlled by one entity or person only, the risk of loss is also wholly tied to this single entity. Besides the single point of failure, it also allocates all the power to spend funds to that single entity, which may not always be desirable. There are use cases that ask for risk and power sharing between multiple entities in relation to the transaction creation process.

A simple approach to split the risk of loss and the power of spending among different entities may seem to split the seed phrase into the desired amount of pieces. However, this should by all means be avoided for the following reasons. First of all, it is hardly avoidable that the entity conducting the split is not exposed to the original seed in its entirety. Second, operating with such a split seed phrase is both inconvenient and insecure as for every transaction the seed must somehow be reassembled. Lastly, it becomes over-proportionally easier for brute-force attackers to find the complete seed phrase once a fragment has been exploited, as stated previously.

[7] For example Blockstream, mempool.space, or many more.

A much better approach to serve the requirement of separation of risk and power in terms of UTXO control are so-called *multi-signature wallets*. This section elaborates on this advanced wallet type.

5.2.1 Definition

The previously elaborated wallet setups, such as paper wallets or HD-deterministic wallets, share the common aspect of requiring only one private information K, e.g. private key or seed phrase, in order to spend UTXOs. Multi-signature wallet setups diverge in this central aspect as they are designed to require more than one K to create valid transactions.

More specifically, multi-signature wallets require M out of N valid signatures, where M denotes the quorum and N the number of feasible public keys to choose from with $M \leq N$. A standard setup is, for instance, a 2-of-3 multi-signature wallet implicating that at least two from three possible private keys are required to sign a transaction in order to make it valid. In the following lines, the M-of-N relationship is also referred to as *threshold* and the N involved key holders as *participants*.

A new transaction becomes valid after at least so many participants have signed it as stated in the threshold. The medium of exchange, in this case, are PSBTs. The PSBT can sequentially be sent to the various participants using insecure communication channels. It allows the co-signers to reside in different geographic locations, as all communications required to complete the transaction can be conducted digitally.

It is essential to understand that the setup and management of a multi-signature wallet involve more different parts than the wallet solutions shown so far. Thus they can be denoted as more complex than single-signature setups. The rationale behind this is that for receiving and (most often) sending funds, the threshold and the xPubs of all participants must be known by any device that crafts the transaction. The signing device must correctly embed the participants' public key information and the threshold to generate a suitable receiving address. Otherwise, the received funds may no longer be spendable using the intended multi-signature scheme. Spending is similarly critical as usually a change UTXO is involved.

Also worth noting is the differentiation of multi-signature wallet setup to the so-called *Shamir's Secret Sharing (SSS)* schema, which was founded by Shamir in 1979. SSS allows the uniform split of a secret into N pieces, which can then be distributed to different parties. In order to restore the secret, T pieces must be provided, whereas it is common practice to have $T < N$. Although these two technologies seem to be similar from a conceptual perspective, i.e. requiring a threshold of participating key or piece holders, they differ fundamentally. Lopp [37] elaborates these differences alongside with reasons to avoid SSS in field of Bitcoin.

5.2.2 Use Cases

The use cases for multi-signature wallet setups are numerous. In practice, however, these use cases are not prevalent due to the advanced nature of this wallet type.

Shared Spending Authority

The most prominent application nowadays is motivated by eliminating sole spending power utilizing, for example, 2-of-3 schemas. Reconsidering the example of company funds. When an employee who used to transact with these funds leaves the company, the funds can simply be secured by revoking this former employee's system access and transaction authority. In terms of Bitcoin funds, revoking access rights is not as straightforward since Bitcoin is a purely informative concept which no one can be excluded from. In other words, the company cannot simply revoke the former employee's knowledge of a seed phrase. Therefore, it is wise to hold the company's funds in multi-signature wallets that require at least two different individuals to spend from them. Figure 5.1 exemplifies this using three different executive roles.

Multi-signature wallets would not only force multiple entities to verify transactions before broadcasting them but also allow for revoking primarily granted signature rights of former employees. The Latter is accomplished by transferring all funds to a new multi-signature wallet setup that no longer includes the public key of the former participant.

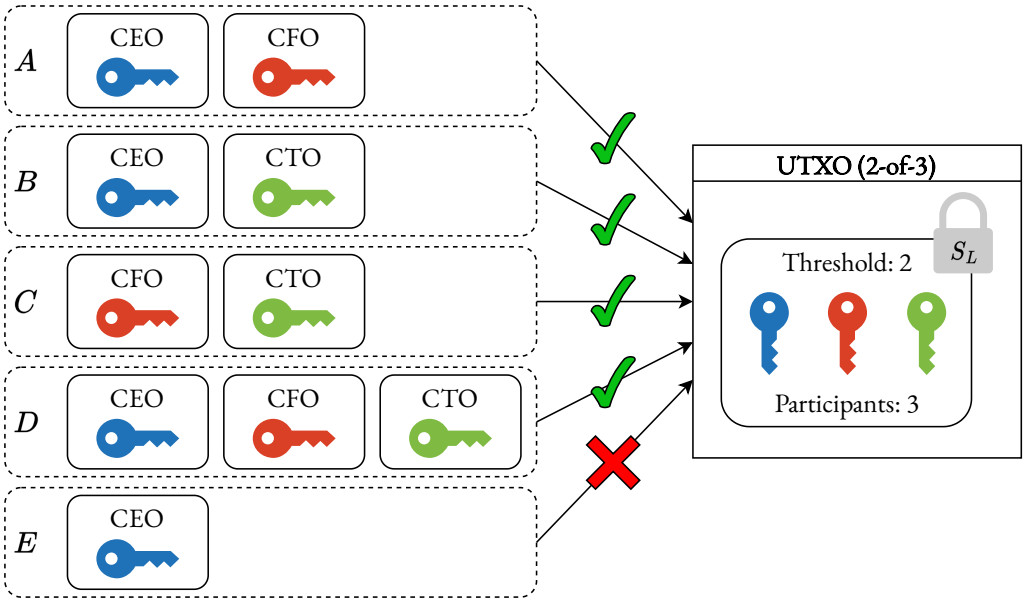

FIGURE 5.1: 2-of-3 Multi-Signature Scheme
Multi-signature wallets allow the creation of UTXOs that are only spendable if the required threshold of signatures is provided to unlock the locking script S_L. The different signers are typically individuals who maintain exclusive knowledge about their private keys. In terms of a 2-of-3 scheme, this would result in four different feasible spending scenarios *A*, *B*, *C*, and *D*. Although *D* is a legitimate spending scenario, it would be slightly less efficient as the minimum of two required signatures is exceeded. Scenario *E*, on the other side, will fail to unlock the spending script of the underlying UTXO as the threshold has not been reached.

Improved Resilience

Also, a field of application for multi-signature schemas lies in Bitcoin's personal use, aiming for higher resilience. For example, a user could set up a multi-signature wallet using multiple devices from different providers to minimize the probability of having a compromised or faulty hardware wallet. The ownership of all keys lies by the same user, but the key generation process was initiated on different independent computing devices. Consequently, these devices can be stored in geographically distant locations. Specifying a threshold of more than one key keeps the funds secure even if a single device storage location is compromised or damaged. A threshold smaller than the total number of participants still allows for complete funds recovery.

Trustless Custodial Services

A more sophisticated use case locates in the area of custodial services. As previously discussed, custodians are entities that manage Bitcoins of their user base through a more guided interaction interface which beginners especially appreciate. However, these services should be used under caution as the sole ownership of the private key is not granted, and users have to trust that their funds are not subject to mismanagement. Combining a multi-signature setup with a *time lock*, i.e. a condition requiring a minimum time to spend a UTXO specified in a future block height, provides an alternative solution that allows for customer support without the perpetual risk of mismanaging funds.

More specifically, the customer locks its funds in a 2-of-2 multi-signature setup where one participant is the customer and the second represents the custodian. A time lock of e.g. 2 years[8] in the locking script will promote the signing power of the custodian to a 1-of-2 signature scheme once the given time is passed. In the meantime, every transaction the customer would like to conduct is mutually signed. After the loss of private keys on the customer's side, the custodian can recover the funds after the given time lock mark has been passed. Suppose this recovery scenario went unused due to successful private key management. In that case, the funds are transferred to a new lock script with an extended time lock just before the anticipated block height is reached. This ensures that the customer remains in complete control and ownership while leaving the custodian with no scope of action until the next block target is reached.

Contrary to the chosen section title, such custodians do, of course, not wholly operate trustless. If a customer has lost his keys and the time lock opens, the custodian receives sole power of attorney over all funds. Of course, this presupposes that the customer trusts in the integrity of the service. However, it is way more trustless than custodial services, which completely deprive their user base of any self-sufficient bitcoin interaction. The used term «*trustless*» intends to emphasize the difference between these two types of custodians in this work.

[8] Since on average every 10 minutes a new block is found, a time lock is indicated using an anticipated block height.

Take-Aways

Although multi-signature wallets are less often used in practice than single-signature setups, they serve various advanced use cases which otherwise must be served with the costs of decreased security. Such use cases include the fragmentation of spending authority among different entities and the improved resilience by utilizing multiple key generation devices or through a minimum of two or more co-signers. Custodial services that only receive full authority over customer funds after a given period represent another use case that is only enabled through multi-signature setups.

5.2.3 Benefits & Pitfalls

The presented use cases demonstrate how multi-signature wallet setups enable a new facet of Bitcoin interaction. Two factors constitute why this wallet type is declared as advanced. One is the modest popularity, the other the various pitfalls associated with its usage.

Therefore, the following lines highlight the most notable advantages resulting from multi-signature schemes and point out the associated pitfalls. Raising awareness will allow the readership to avoid the latter and maximize the potential of this advanced wallet type.

Improved Security

Generally, multi-signature wallets with a threshold of two or more secure the underlying funds significantly. When an attacker manages to compromise any number of participating keys below this threshold, this would still not suffice to steal any funds. The success of such an attack becomes even more unlikely when the individual keys are stored offline in geographically distanced locations. Locations in that context may also refer to people constantly moving around. However, these participants' key storage locations should undergo verification regularly to ensure it is still intact as desired.

A pitfall occurs when all participant keys are stored at the same location. This would immediately eliminate the added security of multi-signature schemas while the higher complexity of such setups remains. Consequently, the underlying funds can then be considered as less secure as they would be in a single-signature wallet setup.

Another pitfall lies in the xPub information that must necessarily be shared in multi-signature setups. As all participants must have knowledge of all xPubs, one should always create an entirely new key pair to participate in such a setup. Using a pre-existing HD wallet would compromise the privacy of the key holder since the provided xPub empowers all other participants to examine the complete transaction history[9].

[9] Assumes that the pre-existing HD wallet follows the default address derivation path.

More Capabilities

Multi-signature wallets open the door for a whole new facet of Bitcoin interaction, which is another significant benefit. Shared authority over funds and, therewith, split risk as well is crucial in business use cases. It was also shown that multi-signature setups enable the compromise between the complete self-management and custodians, i.e. initially declared as trustless custodial services. The demand for such hybrid solution in terms of responsibility will increase with further adoption as not every Bitcoin user will be attracted by the idea of complete self-responsibility of funds.

More capabilities come with the cost of more complexity which in turn introduces more pitfalls. Working with such advanced locking scripts requires a clear understanding of the underlying wallet implementation. Fewer applications handle multi-signature setups correctly due to their lesser-known nature compared to single-signature setups. An erroneously created locking script of a new transaction can lead to an irreversible loss of funds. This may be due to misinterpreted time lock or a falsely generated change address.

Backup Strategy

Creating working backups of multi-signature wallet setups is not accompanied by any benefits. However, it bears a significant pitfall that should at all means be avoided. As mentioned before, such a backup involves more elements than present in a single-signature setup, i.e. in particular the xPubs of all participants and the threshold. Imagine one participant of a 2-of-3 schema, for instance, loses the seed that was used to generate the associated xPub. Although the remaining two participants seem to suffice to spend funds from this wallet setup, they may fail to find spendable UTXOs. The rationale behind this lies in the fact that in order to find spendable funds, all three xPubs must be known by the wallet application. Therefore, it is recommended that each participant, independently of the others, follow a backup strategy that includes not only their own seed but also the xPubs of all other participants.

Take-Aways

Multi-signature wallet setups introduce potent features compared to the more popular single-signature setups. Requiring multiple keys to file a transaction allows storing this keying material in different locations, significantly increasing security. Widened Bitcoin interaction capabilities allow, for example, trustless custodial services or temporally locked funds thanks to time locks.

However, this clearly advanced wallet type comes at the cost of increased complexity. It is essential to understand how wallet implementations handle multi-signature transactions, as otherwise a complete loss of funds may occur. Equally important is a working backup strategy which consists of more elements than conventional single-signature wallet setups. When it is no longer possible to have access to all participating xPubs, it will no longer be possible to find spendable funds.

6 Conclusion

Preceding the outline of different Bitcoin wallets, this work elaborated the fundamentals required to understand this relatively new technology where various subjects from computer science, cryptography, mathematics, and game theory converge. First and foremost, Bitcoin is the name of a communication protocol that implements numerous standards for sharing information among independent participants. Anyone can access, use, and modify it as it resides in the free and open-source domain. The increasingly important free and open-source philosophy is explained and emphasized how crucial it is in Bitcoin.

At the core of this protocol lies the idea of storing and transferring value using asymmetric encryption and decentralized ledgers. After the publication of Bitcoin's first version by a pseudonymized authorship named Nakamoto in 2008, this idea of a trustless, permissionless, borderless, and censorship-resistant value storage and transfer technology has reached popularity. It is to this day continued by a flourishing community.

These prerequisites explained the most critical elements that directly or indirectly relate to the goal of understanding Bitcoin as it allows the readership of different backgrounds to reach a common basis. It includes the definition of general terms often encountered in computer science, such as protocol, network, hash function, public-key cryptography, digital signatures, or various encoding styles.

More Bitcoin-specific term explanations address how any computer device could act as a node in a network to share and consume information about transactions and new data packages attacked to the self-managed blockchain copy. Unspent transaction outputs can be reallocated by providing a suitable unlocking script, most often in the form of a digital signature using an appropriate private key. Wallets have access to such keying material and can thus manage balances by scanning the entire transaction history, i.e. the blockchain, assembling new transactions by unlocking feasible outputs and reassigning them to new owners using a new locking script. The latter most often involves the public key of this new owner. Meanwhile, miners bundle unconfirmed transactions in data packages known as blocks and try to find a suitable block hash value that respects the minimal required leading zeros known as the difficulty target. A newly pronounced next block will be broadcasted over the entire Bitcoin network and immediately verified by every participant.

The central part of this work provides an outline of different wallet types. This includes not only their definitions but also the discussion on the advantages and disadvantages introduced by using a given wallet solution. By doing so, the focus lies on the security- and privacy-preserving interaction with Bitcoin. This work structures the different wallets

according to the individual level of experience in this field, namely beginner, intermediate, and advanced, to facilitate the identification of a suitable wallet for the readership.

Beginners typically encounter two typical Bitcoin interaction means. The first one relates to online accounts. Here, companies provide the complete management of their customer's funds as a service. Doing so, they act as custodians. Customers usually only need a username and password to log in to their service provider's online platform, where they can eventually interact with Bitcoin. However, abstracting complexity and the responsibility of self-accountability from the user comes at the cost of being forced to trust. History has repetitively shown how such providers misuse this trust.

Paper wallets represent another beginner-friendly way of Bitcoin interaction. It describes the idea of possessing a single pair of keys, i.e. private & public keys, that resides in the sole possession of the key owner. Initially, these keys were physically printed on paper. The various methods to generate the keying material include analogue calculations, software tools, or online services. One should always avoid the latter, as the desired exclusive knowledge over the private key can never be guaranteed. In contrast to online accounts, paper wallets represent the first though primitive solution that allows for complete self-custody.

Elaborating on different software and hardware wallets addresses intermediate Bitcoin users. The first one serves as a basis for various wallet derivation techniques. As paper wallets impair the user's privacy by typically only allowing for one single-address, most software wallet implementations utilize (non)deterministic address derivation techniques to overcome this drawback. Hierarchical deterministic wallet derivation follows a specific standard that allows repetitively generating a tree of key pairs rooted in one single secret known as seed phrase. Such a seed phrase consists of twelve or more unambiguous words and can be extended using an extension word.

Hardware wallets represent another intermediate Bitcoin interaction medium. Here, the essential characteristic is that the keying material is stored on a dedicated, temper-proof hardware device that maintains no direct connection to the internet. Such a particular storage medium improves the security of the user's funds significantly and increases the importance of a working backup strategy.

Advanced Bitcoin users will sooner or later be interested in running their self-managed full node. This contributes to the decentralization of the transaction history and further minimizes the dependency on third parties. Since every transaction is self-verified, the need to trust is wholly removed. Additionally, the exclusive communication between someone's wallet with a self-managed Bitcoin full node improves privacy as no intermediate party can witness what is communicated.

The last wallet type outlined concerns multi-signature wallets. It describes the act of locking unspent transaction outputs with a script that requires a minimum of signatures from a predefined set of public keys. A typical example represents a 2-of-3 multi-signature wallet. It not only allows for shared spending authority but also serves as a tool to revoke

someone's participation in such a wallet setup elegantly. Both properties are especially useful in a business context where, for instance, several employees share this authority. Besides this, multi-signature setups are also found in private environments, as multiple keys stored in geographically distanced locations result in higher resilience. However, the potential of this advanced medium of Bitcoin interaction is only beneficial if the user understands its inner workings and knows the importance of a working backup strategy which is more complex in this case.

The development of the Bitcoin protocol is everything but concluded yet. Meanwhile, many side-projects such as blockchain explorers, software wallet implementations, or full node installers have been established and continue to be developed by the community. One of the most dynamic projects in this area is *Lightning*, a second-layer protocol that aims to significantly increase the scalability of Bitcoin transactions. It will also face the expected threats from quantum computing and will eventually have to move to post-quantum cryptography.

In summary, Nakamoto's idea of a peer-to-peer electronic cash system continues to evolve and gain adaption. This work contributes to this process by increasing its accessibility to a readership with different levels of experience and facilitating the secure, privacy-preserving, and self-custody Bitcoin interaction for everyone. Ultimately, the choice of which wallet type to use ultimately rests with the reader. What should always be kept in mind, however, is the wise proverb «*not your keys, not your coins*».

Bibliography

[1] Satoshi Nakamoto. *Bitcoin P2P e-cash paper*. E-mail. The Cryptography Mailing List. Oct. 31, 2008. URL: https://www.metzdowd.com/pipermail/cryptography/2008-October/014810.html (visited on 09/02/2022).

[2] David Chaum. "Blind Signatures for Untraceable Payments". In: *Advances in Cryptology*. Boston, MA: Springer US, 1983, pp. 199–203. ISBN: 978-1-4757-0602-4. DOI: 10.1007/978-1-4757-0602-4_18.

[3] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". White-Paper. Oct. 31, 2008.

[4] Satoshi Nakamoto. *Bitcoin open source implementation of P2P currency*. P2P foundation. Feb. 11, 2009. URL: http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source (visited on 09/02/2022).

[5] Phil Wilson. *Bitcoin Symbol and Logo Origins*. Medium. Mar. 27, 2017. URL: https://medium.com/@_Phil_Wilson_/bitcoin-symbol-and-logo-origins-5d428d40380 (visited on 09/05/2022).

[6] Ludwig Von Mises. *The Theory of Money & Credit*. OCLC: 781455429. Auburn, Alabama: Ludwig von Mises Institute, Dec. 7, 2009. ISBN: 978-1-933550-55-8.

[7] Saifedean Ammous. *The Bitcoin Standard: The Decentralized Alternative to Central Banking*. Hoboken, New Jersey: Wiley, 2018. 1 p. ISBN: 978-1-119-47391-6 978-1-119-47389-3.

[8] Linux & Open Source Annual. *Linux & Open Source Annual 2022: Everything You Need to Master Open Source Software and Operating Systems*. Digital. 7 vols. Linux & Open Source Annual. Future Publishing Ltd, 2022. ISBN: 978-0-203-78790-8.

[9] Sam Williams and Richard M. Stallman. *Free as in Freedom (2.0): Richard Stallman and the Free Software Revolution*. OCLC: 701807052. Boston, MA: Free Software Foundation, 2010. ISBN: 978-0-9831592-1-6.

[10] Gordon Haff. *How Open Source Ate Software: Understand The Open Source Movement and So Much More*. Second edition. OCLC: 1240630758. Berkeley, CA: Apress, 2021. ISBN: 978-1-4842-6800-1.

[11] Open Source Initiative. *The Open Source Definition*. Mar. 22, 2007. URL: https://opensource.org/osd (visited on 09/07/2022).

[12] Axel Metzger, ed. *Free and Open Source Software (FOSS) and other Alternative License Models: A Comparative Analysis*. 1st ed. 2016. Ius Comparatum - Global Studies in Comparative Law 12. Cham: Springer International Publishing : Imprint: Springer, 2016. 1 p. ISBN: 978-3-319-21560-0. DOI: 10.1007/978-3-319-21560-0.

[13] Arun Batchu, Anne Thomas, and Mark Driver. *Hype Cycle™ for Open-Source Software, 2021*. G00747504. Gartner Inc., July 26, 2021. URL: https://more.suse.com/Global_Webpage_Gartner_Open_Source_Report.html (visited on 08/25/2022).

[14] Elaine Barker. *Recommendation for Key Management: Part 1 – General*. NIST Special Publication (SP) 800-57 Part 1 Rev. 5. National Institute of Standards and Technology, May 4, 2020. DOI: 10.6028/NIST.SP.800-57pt1r5. URL: https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final (visited on 09/11/2022).

[15] Elaine Barker and William Barker. *Recommendation for Key Management: Part 2 – Best Practices for Key Management Organizations*. NIST Special Publication (SP) 800-57 Part 2 Rev. 1. National Institute of Standards and Technology, May 23, 2019. DOI: 10.6028/NIST.SP.800-57pt2r1. URL: https://csrc.nist.gov/publications/detail/sp/800-57-part-2/rev-1/final (visited on 09/11/2022).

[16] Elaine Barker and Quynh Dang. *Recommendation for Key Management, Part 3: Application-Specific Key Management Guidance*. NIST Special Publication (SP) 800-57 Part 3 Rev. 1. National Institute of Standards and Technology, Jan. 22, 2015. DOI: 10.6028/NIST.SP.800-57pt3r1. URL: https://csrc.nist.gov/publications/detail/sp/800-57-part-3/rev-1/final (visited on 09/11/2022).

[17] Shayan Eskandari et al. "A First Look at the Usability of Bitcoin Key Management". In: (Feb. 12, 2018). DOI: 10.14722/usec.2015.23015. URL: https://arxiv.org/abs/1802.04351v1 (visited on 09/11/2022).

[18] Chuck Easttom. *Modern Cryptography: Applied Mathematics for Encryption and Information Security*. New York: McGraw-Hill Education, 2016. 1 p. ISBN: 978-1-259-58809-9.

[19] Jean-Philippe Aumasson and Matthew D. Green. *Serious Cryptography: A Practical Introduction to Modern Encryption*. OCLC: ocn986236585. San Francisco: No Starch Press, 2017. 282 pp. ISBN: 978-1-59327-826-7.

[20] Jean-Phillippe Aumasson. *Crypto Dictionary: 500 Tasty Tidbits for the Curious Cryptographer*. San Francisco: No Starch Press, 2021. 145 pp. ISBN: 978-1-71850-140-9.

[21] National Institute of Standards and Technology (NIST). *Secure Hash Standard (SHS)*. Federal Information Processing Standard (FIPS) 180-4. U.S. Department of Commerce, Aug. 4, 2015. DOI: 10.6028/NIST.FIPS.180-4. URL: https://csrc.nist.gov/publications/detail/fips/180/4/final (visited on 10/23/2022).

[22] Abhijit Das and C. E. Veni Madhavan. *Public-Key Cryptography: Theory and Practice*. Upper Saddle River: Pearson Education, 2009. 562 pp. ISBN: 978-81-317-0832-3.

[23] C. E. Shannon. "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27.3 (July 1948). Conference Name: The Bell System Technical Journal, pp. 379–423. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[24] Andreas M. Antonopoulos. *Mastering Bitcoin: Programming the Open Blockchain*. Second edition. Sebastopol, CA: O'Reilly, 2017. 371 pp. ISBN: 978-1-4919-5438-6.

[25] Andreas M. Antonopoulos, Olaoluwa Osuntokun, and René Pickhardt. *Mastering the Lightning Network: A Second Layer Blockchain Protocol for Instant Bitcoin Payments*. First edition. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly, 2021. 438 pp. ISBN: 978-1-4920-5486-3.

[26] Paul C Van Oorschot. *Computer Security and the Internet: Tools and Jewels From Malware to Bitcoin*. Cham, Switzerland: Springer, 2021. ISBN: 978-3-030-83411-1. URL: https://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=6768682 (visited on 05/29/2022).

[27] Vishal Thakur. *How Many Atoms Are There In The Universe?* Science ABC. May 12, 2022. URL: https://www.scienceabc.com/nature/universe/how-many-atoms-are-there-in-the-universe.html (visited on 10/13/2022).

[28] C. P. Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Ed. by Gilles Brassard. Lecture Notes in Computer Science. New York, NY: Springer, 1990, pp. 239–252. ISBN: 978-0-387-34805-6. DOI: 10.1007/0-387-34805-0_22.

[29] Stefano Bistarelli, Ivan Mercanti, and Francesco Santini. "An Analysis of Non-standard Bitcoin Transactions". In: *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. 2018 Crypto Valley Conference on Blockchain Technology (CVCBT). June 2018, pp. 93–96. DOI: 10.1109/CVCBT.2018.00016.

[30] Ben Charoenwong and Mario Bernardi. *A Decade of Cryptocurrency 'Hacks': 2011 – 2021*. Rochester, NY, Oct. 1, 2021. DOI: 10.2139/ssrn.3944435. URL: https://papers.ssrn.com/abstract=3944435 (visited on 10/01/2022).

[31] Lane Brown. *What Happened to FTX?* Intelligencer. Nov. 19, 2022. URL: https://nymag.com/intelligencer/article/sam-bankman-fried-ftx-bankruptcy-what-happened.html (visited on 11/21/2022).

[32] Makena Kelly. *Coinbase says it halted more than $280,000 in bitcoin transactions during Twitter hack*. The Verge. July 20, 2020. URL: https://www.theverge.com/2020/7/20/21331499/coinbase-twitter-hack-elon-musk-bill-gates-joe-biden-bitcoin-scam (visited on 11/04/2022).

[33] Rick D and Kyle Baird. *Freedom vs Protection: Should Crypto Exchanges Censor Transactions?* BeInCrypto. July 22, 2020. URL: https://beincrypto.com/freedom-vs-protection-should-crypto-exchanges-censor-transactions/ (visited on 10/01/2022).

[34] dabura667. *Can you create a Bitcoin address manually?* Bitcoin Forum. Jan. 10, 2015. URL: https://bitcointalk.org/index.php?topic=919430.msg10101533#msg10101533 (visited on 01/10/2015).

[35] Roger Huang. *How Bitcoin And WikiLeaks Saved Each Other*. Forbes. Section: Crypto & Blockchain. Apr. 25, 2019. URL: https://www.forbes.com/sites/rogerhuang/2019/04/26/how-bitcoin-and-wikileaks-saved-each-other/ (visited on 10/09/2022).

[36] Adi Shamir. "How to Share a Secret". In: *Communications of the ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/359168.359176. URL: https://dl.acm.org/doi/10.1145/359168.359176 (visited on 12/09/2022).

[37] Jameson Lopp. *Shamir's Secret Sharing shortcomings*. Casa Blog. Oct. 17, 2019. URL: https://blog.keys.casa/shamirs-secret-sharing-security-shortcomings/ (visited on 12/09/2022).

List of Abbreviations

ASCII	American Standard Code for Information Interchange	9
BTC	Bitcoin	14
BIP	Bitcoin Improvement Proposal	18
DHKE	Diffie-Hellman Key-Exchange	8
DLP	Discrete Logarithm Problem	11
ECDSA	Elliptic Curve Digital Signature Algorithm	8
FLOSS	Free/Libre and Open-Source Software	4
FOSS	Free and Open-Source Software	4
GUI	Graphical User Interface	22
HD	Hierarchical-Deterministic	31
HTTP/S	Hypertext Transfer Protocol/Secure	6
ICT	Information and Communication Technologies	2
KYC	Know Your Customer	19
NIST	National Institute of Standards and Technology	5
P2PKH	Pay to Public Key Hash	11
P2SH	Pay to Script Hash	11
P2TR	Pay to Taproot	11
P2WPKH	Pay to Witness Public Key Hash	11
PSBT	Partially Signed Bitcoin Transaction	36
QR	Quick Response	12
RAM	Random Access Memory	41
RNG	Random Number Generator	24
RSA	Rivest-Shamir-Adleman	8
SegWit	Segregated Witness	12
SBC	Single-Board Computers	41
SHA	Secure Hash Algorithm	7
SHS	Secure Hash Standard	7
SSS	Shamir's Secret Sharing	45
UTXO	Unspent Transaction Output	13
xPub	Extended Public Key	32

List of Figures

List of Tables