SoK: Web3 Recovery Mechanisms

Panagiotis Chatzigiannis¹, Konstantinos Chalkias², Aniket Kate³, Easwar Vivek Mangipudi⁴, Mohsen Minaei¹, and Mainack Mondal⁵

¹ Visa Research {pchatzig,mominaei}@visa.com
 ² Mysten Labs kostas@mystenlabs.com
 ³ Purdue University / Supra Research aniket@purdue.edu
 ⁴ Supra Research e.mangipudi@supraoracles.com
 ⁵ IIT Kharagpur mainack@cse.iitkgp.ac.in

Abstract. Account recovery enables users to regain access to their accounts when they lose their authentication credentials. While account recovery is well established and extensively studied in the Web2 (traditional web) context, Web3 account recovery presents unique challenges. In Web3, accounts rely on a (cryptographically secure) private-public key pair as their credential, which is not expected to be shared with a single entity like a server owing to security concerns. This makes account recovery in the Web3 world distinct from the Web2 landscape, often proving to be challenging or even impossible. As account recovery has proven crucial for Web2 authenticated systems, various solutions have emerged to address account recovery in the Web3 blockchain ecosystem in order to make it more friendly and accessible to everyday users, without "punishing" users if they make honest mistakes. This study systematically examines existing account recovery solutions within the blockchain realm, delving into their workflows, underlying cryptographic mechanisms, and distinct characteristics. After highlighting the trilemma between usability, security, and availability encountered in the Web3 recovery setting, we systematize the existing recovery mechanisms across several axes which showcase those tradeoffs. Based on our findings, we provide a number of insights and future research directions in this field.

1 Introduction

Blockchain-based applications such as cryptocurrencies, are typically associated with a private-public key pair to authenticate a user transaction. This requires the user to obtain and maintain the private key securely. While requiring this key prevents unauthorized transactions, a potential loss of the key can lock out the user of his/her accounts and, therefore, permanently lose access to all associated funds and assets. The decentralized nature of the blockchains implies that there is no single entity with the power to restore access. From the early days of Bitcoin, there have been numerous stories, some quite dramatic, where users lost access to their funds as they could no longer access their keys for various reasons. In Web2 (or Web 2.0), users often rely on service providers for account recovery. If users forget their password or lose access to their account, they typically have to go through a service-led account recovery process. This process often involves identity verification, security questions, or email verification.

As Web3 is building upon the core concepts of decentralization and openness, this centralized process of account recovery is at direct odds. For Web3 applications over blockchains, users need to have more responsibility to safeguard their accounts. If they lose their private keys, in most cases no central authority or service provider can help them recover their account or their funds. If an adversary steals their key and transfers funds away, there are no mitigations as in credit card fraud protection schemes.

Some initial research efforts have been directed toward account recovery in the blockchain setting. Cryptocurrency wallets are emerging that offer some form of built-in recovery support in an attempt to give the user some confidence that the account or its funds will still be available even if the private key is lost. The idea is to generally make the Web3 world more "forgiving" and bring it closer to the average user's experience of mainstream Web2 payment systems.

Asset and account loss types. "Losing" the private key is one of the obvious cases where a user will also lose access to an account and all of its associated funds. Such cases include losing or destroying the medium of its storage, accidental deletion, hardware failure, viruses and malware etc. [12] A user passing away is also considered a potential case of asset or account loss, as in cryptocurrencies there is no built-in way of transferring funds to the heirs' custody unless the user has taken the prior steps necessary to transfer the private key to the heirs upon his/her death [19].

A user might also lose funds when mistyping the recipient's address, with the assets being transferred under an account for which a private key does not exist [10,14]. Unless specific countermeasures are taken by the application to prevent such cases from happening (e.g., checksums [31] or QR codes), these instances are effectively equivalent to key loss.

We note that a recent work by Maram et al. [50] considers four possible states of a credential or key: *safe* (only the user knows it), *leaked* (the user and an attacker know it), *stolen* (only an attacker knows it) and *lost* (neither the user nor the attacker knows it). In this work, we only consider recovery for keys that are in a "lost" state, and we consider cases where keys are in a "leaked" or "stolen" form as out of scope, as the recovery approaches discussed in this paper are not generally applicable. However, under certain circumstances and special designs, some recovery methods (e.g., key rotation) might still work for leaked keys, as we discuss later.

Our contributions. In this paper, we perform a systematization of recovery solutions in the Web3 domain. We provide definitions of the various characteristics of these solutions across different dimensions; we categorize them based on these properties by highlighting the subtle differences. Finally, through our systematization methodology, we identify several gaps in the literature and provide interesting insights for the future research directions.

The rest of this paper is organized as follows. In Section 2, we provide definitions for the primitives and cryptographic background used in our paper and a general overview of existing recovery methods in the Web3 space. In Section 3 we present the systematization methodology we use in our work. In Section 4, we review the most prominent recovery solutions that (to the best of our knowledge) exist at the time of writing of this paper and categorize them based on the systematization methodology we defined previously. We omit solutions that are identical or with fewer functionalities in terms of authentication and recovery compared to those we consider. In Section 5, we discuss the various insights and research gaps we discovered based on our systematization methodology as well as our broader observations. In Section 6, we briefly overview some additional cryptocurrency wallets that provide basic recovery mechanisms and related works more broadly related to the Web3 recovery challenges. We conclude our paper in Section 7 with a summary of proposed research directions in this field.

2 Preliminaries and Existing Recovery Methods

2.1 Definitions and background

We now provide informal definitions of the various elements and primitives used in our paper. We provide formal definitions in the Appendix A.

Secret Sharing. A Threshold Secret Sharing scheme [36] is a mechanism to generate shares of a secret sk among n parties such any t+1 or more parties can reconstruct the secret value. However, no set of parties of size less than or equal to t can recover the secret.

Threshold signatures. A digital signature scheme lets a user authenticate a message m by signing it using a secret key sk corresponding to a public key pk. A threshold signature scheme [54,30] lets a set of parties generate partial threshold signatures such that the aggregated signature, generated from any t + 1 or more partial signatures, can be verified by the group public key. Here, the secret key required for the signature generation is split among the set of parties generating the threshold signature.

Multi-signatures. In a Multi-signature scheme [30,40], each of the *n* parties has an independent secret key - public key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ and generates an independent signature $\sigma(\mathsf{sk}_i, m)$. These signatures are collected (and concatenated) to form a Multi-Signature. Any threshold access structure can be realized using the MultiSig scheme. The MultiSig is considered valid if at least t + 1 of the parties produced valid signatures on the message *m* being signed.

Group signatures. A group signature [27] allows any group member to sign anonymously on behalf of the group of size n. Any verifier can check that a group member has signed the message but can not identify the signer unless revealed. **Secure multi-party computation (MPC).** MPC consists of the cryptographic methods enabling parties to jointly compute a function, where each party provides a private input [33,58]. One of the main requirements in MPC is that each party's input is not revealed to anyone, but also no information related to that input is leaked due to the protocol execution, except the information that can be inferred by the output.

MPC Wallets. In the blockchain setting, MPC wallets are generated by two or more parties⁶ which generate the public key associated with the wallet in an interactive fashion, while retaining their individual keys (or shares) as their private information, which are then used to sign messages (e.g., a transaction). For the purposes of our paper, we overload the definition to include wallet that make use of any of the cryptographic signature schemes mentioned above, i.e. any wallet which is controlled by two or more private keys.

2.2 Recovery methods

We now overview, to the best of our knowledge, some of the most prominent methods for recovering accounts in the Web3 setting. For simplicity, we assume that an account is reflected by the account owner's public key pk, although accounts in most blockchain applications are reflected using additional operations on top of pk (e.g., a hashed public key). Under normal circumstances, the account holder uses the private key sk associated with pk to create transactions on the blockchain (e.g., create a signature $\sigma(m)$ where m is an instruction to perform a value transfer from the account holder's balance or execute some smart contract operation which only the account holder is authorized to perform). However, as secure key management in blockchains remains an issue (especially for the average user), there are cases where the account holder loses access to the private key, as discussed in the "loss types" previously in Section 1. In the following paragraphs, we provide an initial overview of ways of recovery in blockchains.

Off-chain backup and recovery. The first and most intuitive method available to enable account recovery is to back up **sk** into a secondary medium and recover it on demand. However, as the **sk** is a sensitive piece of information, the backup must be performed securely to prevent attackers from accessing it, compromising the account and draining its assets.

The easiest way to perform such a backup is to copy sk to an offline device (e.g. a USB thumb drive) and store it somewhere safely. This provides a method to move a key back from the "lost" state to the "safe" state; however, it cannot move it back from the "leaked" or "stolen" states. Also, a private key in plaintext (even in offline devices) still raises security concerns. Encrypting the private key either using a software encryption scheme or with a hardware device is still problematic, as the user will have to manage an additional key, while the backed-up key is still susceptible to medium destruction or loss.

A more elegant way of performing a backup in this manner is making use of cloud storage services. Such services are widely used today by the majority of the user base, which makes using a cloud service for key backup a reliable and user-friendly solution. However, despite the security measures employed by cloud providers such as two-factor authentication (2FA), there's still a major concern

⁶ Note that in our setting, a single user can control multiple "parties".

about the backed-up key being compromised with devastating consequences to the account. Encrypting the plaintext key under another key (sometimes referred to as "Key Encryption Key" or KEK), mitigates this risk. However, as before, there's still a problem with managing the KEK. Therefore, some wallet providers [1] natively offer hosting services for the KEK and only help with decrypting sk after verifying the user's request with 2FA methods. In this approach, the security risk is mitigated as an attacker must successfully compromise both the user's cloud provider and the wallet service provider, which offers a good tradeoff between security and user-friendliness. Also, the recovery process is performed off-chain, and the user can regain access to the account without additional delays or costs. The main drawback of this method is that it is proactive, meaning that the user must "plan ahead" and create those backups ahead of time.

Also, instead of utilizing a single KEK, sk can be broken into shares, and then storing the shares into different locations (e.g., some combination of different cloud providers, wallet providers, and/or offline storage). This "key share distribution" approach is essentially an extension/upgrade of using a single KEK, enabling the user to design a customized recovery plan.

Use of seed phrases. As an alternative off-chain backup method, seed phrases, typically consisting of a sequence of random words, provide a user-friendly way of key backup while providing sufficient entropy to create a private key deterministically. Essentially, the user needs to record a small number of words (usually 12) on paper instead of managing cloud services, hardware devices, or additional keys, as in the backup methods discussed before. However, the user needs to keep them somewhere safe, as anyone learning those words is equivalent to a private key compromise. Note that as other methods based on mnemonics such as passwords, PINs or security questions do not provide sufficient entropy to generate a key, these are exclusively used in the context of access control of the wallet app or logging in to the server of a wallet provider as additional security layers. Therefore, such mnemonics cannot be used as a recovery method in a key loss, and recovering access to an account that requires the knowledge of those mnemonics as an extra layer of security is not within the scope of our paper.

Pre-signed transactions. A user could pre-sign transactions which send some or all of the account's funds to a secondary account, and store them in an offline medium. This would work in case of key loss as a fallback mechanism to recover the account's funds [56], however, it is cumbersome for the user and creates a burden to manage the secondary account keys. Also, this method might not work for unspent transaction output (UTXO)-based systems, as the unspent outputs used at the time of pre-signing might be spent later.

Smart contract recovery through guardians. Alternative recovery methods can be created by utilizing the power of smart contracts. One such way is through the use of guardians⁷, which are designated parties with the power to help the user perform account recovery when needed, but without having any

⁷ In the Web3 space, recovery using guardians is sometimes referred to as "social" recovery.

other powers over that account, such as signing messages or spending funds. They are normally hand-picked by the user (e.g., reputable service providers or reliable close friends or relatives), giving the user confidence that they will follow the recovery protocol when asked to help. However, some wallet providers put themselves in a default guardian role to increase user-friendliness. This removes the requirement of proactive action by the user while still providing the option to add (or remove) additional guardians [1]. Naturally, for such recovery methods to work, they require a smart contract that orchestrates all account operations, as the contract provides advanced functionalities needed to assign other parties to an account with special roles. Therefore, these methods are not applicable to standard accounts or in blockchains that do not support smart contracts.

Each of those guardians, denoted by \mathcal{G} , has its public key reflected in the smart contract as $\mathsf{pk}_{\mathcal{G}}$. Then, the smart contract code enables \mathcal{G} to perform special recovery operations, e.g., rotate the account key after the user has generated a new (pk, sk) pair. This operation might be associated with a timed delay enforced by the contract, which lets the user cancel a potential malicious recovery operation within that period e.g., by signing a special "cancel recovery" transaction. Also, the account holder has the power to add, remove or replace guardians if he/she desires, and initiating the recovery operation would require a pre-determined threshold of the guardians set (e.g. over 50%). This essentially mitigates the risk of a rogue guardian trying to steal the user's assets, as the user, after careful planning, would have picked those guardians with diverse characteristics. Naturally, changes to the guardian set or any other logic changes to the smart contract require a digital signature using the account's key.

The main advantage of a smart contract-based recovery (such as using guardians) is that it can be *reactive*, meaning it might not require the user to take any preparatory steps beforehand and can be invoked when the user loses sk. This recovery might help even when the key is in a "leaked" state, for example, when the funds have not yet been fully drained from the user's account (e.g., because of restrictions in the smart contract) and the user becomes aware (or simply suspects) that the key has already been leaked. In this case, the user might ask a guardian to rotate the keys, invalidating the compromised sk.

The disadvantages are that it only applies to accounts associated with or controlled by smart contracts, and the operation is performed on-chain, incurring gas costs. Such an on-chain operation is publicly visible, providing an adversary with ways of exploiting such information. Also, its security and user-friendliness characteristics are not ideal, as the risk of malicious guardians increases the attack surface. The user must either actively monitor the blockchain for unauthorized recovery attempts or rely on third-party services that notify the user (e.g., by email) if a recovery attempt is detected during the enforced time delays.

Smart contract guardian-less recovery. Some methods, such as KELP [29], also based on smart contracts, do not involve guardians at all but rely entirely on the smart contract to perform the recovery operation on behalf of the user. Essentially, the contract lets *anyone* make a recovery claim for some account

(instead of relying on pre-determined guardians), and on such claim receipt, the contract follows a designated protocol to complete the recovery process.

Such a method generally inherits the advantages and disadvantages of the previous one, which involves guardians - it is reactive, as no prior action is required by the user, but requires an account associated with a smart contract, is publicly observable, and incurs paying gas costs. However, parties with special powers are no longer required, and the user does not need to place trust in anyone except the smart contract itself. On the other hand, the time delay will typically be longer than those with guardians, as the trustless model needs to mitigate targeted recovery attempts, e.g., when an account holder is known to be "off the grid" by the attacker. We discuss KELP in more detail in Section 4.

3 Systematization methodology

As in most applications in technology, there is a natural tension between *us-ability* and *security* since hardening security typically comes at the expense of the user's experience. In the Web3 recovery setting, (lack of) security captures an adversary's ability to control an account or its associated assets. However when introducing recovery mechanisms in this setting, an additional property that needs to be considered is *availability*, which captures the user's ability to invoke and complete the recovery functionality successfully when needed.

A trilemma for the recovery mechanisms in wallets. We observe an interesting trilemma among the properties, where only two of the three of usability, strong security, and high availability are being satisfied for any existing recovery mechanism. We informally define these properties and expand on them below.

Usability: A usable system should meet the requirements of the user and should be easy to use [45,46]. However, the nature of non-custodial wallets, where users have control over private keys (and naturally bear a large burden of managing/recovering the keys), often requires complex mechanisms for key recovery, affecting usability. For instance, users may need to manage seed phrases or multiple keys for recovery, which is similar to managing multiple authentication secrets (some of which might be rarely used) and shown to be less usable (due to cognitive burden) [39,37,28]. Similarly, setting recovery mechanisms with the help of two different servers/devices is more usable for users than setting it on hundreds of devices which increases the cognitive burden and possibility of error.

Security: A secure key recovery system should ensure that only the legitimate user can recover their keys (i.e., resistant to any unauthorized recovery). However, other attack vectors should also be considered, which we discuss in detail later in section 3.7, namely denial of recovery, key compromise attacks, and privacy-related threats. Therefore, we consider a recovery system secure if it is resistant against the attack vectors we consider as a whole. Consequently, achieving security often requires complex cryptographic techniques or authentication-level techniques such as multi-factor authentication, which can reduce usability. On the other hand, making key recovery too simple or accessible could make it easier for malicious parties to perform such attacks.

Availability: The key recovery system should be accessible whenever users lose their keys. However, some secure key recovery methods, such as multi-signature schemes or recovery using guardians, depend on the availability of other parties or multiple keys. If these are unavailable, the user may be unable to recover their keys when needed.

To better illustrate the properties and the trilemma between them, we consider a (t, n)-guardian-assisted recovery scheme, where the user needs to select n guardians from a pool of servers (or a pool of social contacts), and at least t guardians are needed to perform a recovery operation.

- Usable and Available (and not secure): A guardian recovery scheme with n = 6 and t = 2 is usable (as the user only needs to setup the recovery with 6 servers) and available (since only 2 servers are required to help the user when required). However it is not very secure because an adversary only needs to corrupt 2 guardians to steal the user's funds.
- Secure and usable (and not available): A guardian recovery scheme with n = 6 and t = 4 is equally usable as before, as the user needs to perform the same proactive tasks. While it is more secure (as the adversary needs to corrupt more servers simultaneously), it is less available (as the user now will require the help of 4 servers when needed).
- Secure and Available (and not usable): If we now consider a scheme with n = 12 and t = 4, it is equally available as the first (assuming the probability of each server crashing or otherwise unavailable remains the same) and as secure as the second (as the adversary needs to corrupt the same amount of servers). However it is less usable, since the user needs to perform the more difficult proactive task of setting up recovery with 12 servers, as finding more guardians with the same high degree of security and availability can be hard in practice.

With all previous examples showcasing the challenges the design of a recovery system must address, we choose to systematize our findings across the dimensions outlined in this section, which highlight the tradeoffs of the usability vs. security vs. availability trilemma.

3.1 Proactive vs. Reactive

The primary distinction among recovery approaches lies in the requirement for prior user action. Approaches like off-chain backups inherently necessitate *userinitiated* actions beforehand, making it essential for users to be proactive. These methods won't be available if users haven't followed the procedure of backing up keys or shares as stipulated by the wallet. In contrast, reactive methods do not necessitate prior user actions and offer recovery options on-demand. The latter is generally preferred, particularly since most users prefer straightforward and expedient wallet setups. In other words, reactive solutions by definition offer the best usability. Based on the above explanation, we present the following definitions:

Definition 1. A recovery method is classified as "proactive" or "off-chain" if it mandates user-initiated actions before the occurrence of account or key loss.

Definition 2. A recovery method is classified as "reactive" or "on-chain" if it does not require any prior user-initiated actions but relies solely on systeminitiated actions.

One could argue that all recovery methods exhibit proactive elements, given that even methods labeled as "reactive" involve some *system-initiated* actions, with the broader system (like the blockchain or smart contract) proactively acting on behalf of the user. In this context, the only truly "reactive" methods would resemble hard forks or the DAO (as discussed in section 6), which rely on mass user votes to determine recovery outcomes.

However, since our focus is primarily on the user's perspective and needs, we still classify methods requiring prior *system-initiated* actions as "reactive." This classification aids users in comprehending their level of involvement and responsibility in securing their assets.

Lastly, we do not view any user-required actions involving access-control mechanisms during account setup, such as passwords or PINs (discussed in Section 2.2) as a "user-initiated" action for recovery purposes. As these are an "everyday" method of accessing the key using additional security layers, we consider these as orthogonal to our work.

3.2 Account recovery vs. Funds recovery

In cryptocurrency systems, in the majority of cases, a user's primary objective is to safeguard the custody of the funds associated with their accounts, ultimately aiming to recover the associated value in case of a loss. However, the recovery process may not always entail the restoration of the initial account, typically represented by the public key. For instance, consider a scenario where a smart contract is programmed to make periodic deposits to a specific account address. In such cases, merely recovering funds from the account may prove insufficient as all subsequent deposits may be lost.

This limitation arises depending on the chosen recovery method and becomes especially pertinent when the account in question has external properties or logic associated with it. This complexity becomes even more apparent in systems like Libra and Aptos [22,26], where accounts are equipped with features such as key rotation and delegated withdrawals, making the recovery process more intricate than a straightforward funds retrieval.

3.3 Standard wallet accounts vs. Smart contract accounts

A standard wallet account⁸ only requires a signature (or a set of signatures) generated by a private key (or multiple keys) in order to be to be able to spend the

 $^{^{8}}$ In the Ethereum ecosystem these are referred to as "externally-owned accounts".

funds associated with that account. However, smart contract accounts are not limited to the functionality above, but add programmability features that can control the funds with arbitrary logic. Examples of such accounts are Ethereum's account abstraction proposal [32] or delegating withdrawal capabilities in Libra/Aptos [22,26]. Therefore as a natural consequence, smart contract accounts offer more sophisticated recovery capabilities as discussed in Section 2.2.

3.4 Recovery using guardians

In certain types of wallets, the client has the option to designate 'Guardians'. Guardians in crypto wallets are individuals or entities that are trusted by the wallet holder to help them recover their assets if they lose their private keys. The client may choose these guardians based on reputation from an available pool. However, the client may also choose parties she trusts as guardians, possibly from her social circle, like friends and family. The guardians may or may not know the identities and existence of other guardians. They perform the asset recovery on behalf of the client in case of key loss. We view this trust-based social recovery as a special case of recovery using guardians.

3.5 Multi-key recovery

For specific varieties of wallets, such as Multi-Party Computation (MPC) wallets or those utilizing multi-signatures or threshold signatures, the authentication process for issuing a transaction does not rely on a single key. Instead, it requires two or more keys. In such scenarios, there may be instances where only one key is lost while the other keys remain accessible. In these situations, the recovery process can be initiated using the remaining keys to replace the lost key. This recovery procedure can be executed either directly within the cryptographic scheme [27] or by utilizing smart contract functionalities.

3.6 Key-custody

In general, the term "custody" in the Web3 space refers to the level of (lack of) control the user has over his/her private key(s). The general consensus in the Web3 community is that a "custodial" wallet refers to a wallet in which the private keys are not held by the user but by a third party, managing the keys on the user's behalf⁹[6,5]. However, with the advent of wallets that are co-managed by the user and a third party using primitives outlined in Section 2.1, we observe irregularities and broad interpretations of the terms frequently encountered in the space such as "non-custodial" or "self-custodial". Mostly, these two terms are being used interchangeably [7,8,9,20], implying that the user has full control of the account. It is without any external party existing with the power to refuse a transaction, but on the other hand without providing any recovery options in case of key loss. However, in recent proposals e.g. the

⁹ Another terminology encountered is "user-controlled" and "developer-controlled" [4].

MPC wallet from Coinbase [47], we observe that these terms became overloaded, with the "self-custodial" wallet definition being used even if an external service provider needs to co-sign the user's transactions, with the "self" term referring to the ability of the user to remove the service provider entirely from the wallet scheme if desired, while the entire wallet is referred to as "Coinbase-aided".

Having observed these inconsistencies in the Web3 ecosystem, we provide the following definitions which we use for our systematization:

Definition 3. A wallet or Web3 account is called "custodial" if all private keys are managed by an external service provider acting on the user's behalf.

Definition 4. A wallet or Web3 account is called "self-custodial" if all private keys are managed by the user, and no single external service provider has the ability to censor or prevent the user's transactions at any point.

Definition 5. A wallet or Web3 account is called "non-custodial" if the private keys are distributed between the user and an external service provider which retains the ability to have some form of control over the user's account.

We use different definitions for the terms "non-custodial" and "self-custodial" to make the distinction of account control explicit.

3.7 Attack vectors

Designing a recovery mechanism for web3 accounts needs careful design, as an adversary might take advantage of them and compromise the user's account. In general, wallets that are augmented with some recovery functionalities are susceptible to the following attack vectors:

- 1. Unauthorized recovery: An adversary might attempt to invoke a recovery of a victim's account with the goal of taking control of the account or transferring its assets to an adversarial account. Note that this attack vector includes cases where the adversary has corrupted one or more recovery agents who belong to the set of authorized agents (if the recovery scheme includes such an approach).
- 2. **Denial of recovery**: While a recovery agent might not act maliciously and actively attempt to control a user's account or its funds, an agent might refuse to cooperate when needed and extort the user for money.
- 3. Key compromise attack: In web3 accounts that have distributed keys (e.g., threshold or MPC wallets), an adversary can take control of one or all of the participating parties to compromise the key or extract secret information [48]. If the adversary compromises a set of devices less than the threshold required for the full recovery of the secret information, the threshold for further compromise is reduced.

3.8 Privacy-related threats

The fact that an address owner is engaging with recovery solutions and related software can leak some private information which can sometimes be indirectly exploited. Some notable examples include:

- 1. **Recovery in progress:** If an adversary becomes aware that a user has initiated a recovery process, they could exploit this information to carry out malicious activities. For instance, they could launch social engineering attacks or attempt to outpace the user in the recovery protocol itself, a tactic known as front-running. This underlines the importance of maintaining confidentiality during the recovery process to mitigate potential security risks.
- 2. Backup structure: Some designs reveal the exact key structure of an account, leading to the extraction of information on the wallet software backing an account. An example is plain or smart contract-based multi-sig accounts exposing the number of keys involved. Interestingly, one of the main advertised features of threshold MPC wallets is that addresses are indistinguishable from regular single-key accounts, and hence, they defend against the aforementioned structure leak of multi-sig solutions. However, even with threshold MPC wallets, it has been reported that some implementations divert from the official signature scheme standards, which can break this indistinguishability property. An example includes the case of ed25519 threshold wallets randomizing their signatures on retries; against the EdDSA standard recommendation (that most plain-key ed25519 implementations follow), which suggests deterministic signatures over the same message [34].

4 Existing solutions

In this section we review the existing solutions for account recovery in a blockchain setting, highlighting their properties based on our definitions provided previously. We distinguish them between recovery proposals and wallets, with the latter category being commercial software implementations.

4.1 Recovery proposals

EIP2429 [53] was among the first solutions to enable account recovery in the Ethereum cryptocurrency. This proposal enables the user to select a private list of Ethereum addresses (i.e., "guardians") which can authorize a call protected by a secret only known to the user. Such a secret, which is never revealed on chain, but hashed with a nonce, can be based on the user's biometrics, personal information or a password, and its intention is to prevent attempts of malicious guardians to take over the user's account without his or her permission. The guardian list is also hidden in a Merkle tree and is only revealed when recovery is invoked. This is intended to prevent collusion between guardians.

ERC4337 [32] while not primarily focused on the recovery aspect, enables users to program Ethereum wallet recovery using guardians.

KELP [29] provides a way to recover funds using a smart contract. It is a reactive method, therefore the user doesn't need to take any action proactively. The most prominent feature of this method, however, is that there are no "guardians" involved, and anyone can invoke a recovery request on any existing account. Therefore KELP needs to mitigate 2 major attack types: First is "front running", where an attacker might see a pending recovery transaction and front-run it by invoking another recovery first, and a "random testing", where the attacker might make recovery attempts for a large number of random accounts.

Therefore the smart contract protocol is split into three phases, a commit, reveal, and challenge phase, with two separate time delay periods in between, each associated with a fee as collateral. The first is used to publish a commitment to a recovery request from one account to another (which prevents front-running attacks). The key assumption however is that the account owner is the first to realize that the key is lost, otherwise an attacker can still publish a recovery request commitment and front-run the account owner. The second period is intended to provide the opportunity for a legitimate account owner to challenge the recovery request. More specifically, similar to the methods involving guardians, a user can "challenge" an unauthorized recovery attempt, where either the attacker would simply forfeit the collateral or the user would claim the collateral amount and credit it towards his/her account. This mechanism acts as a disincentivize towards random testing attacks. We point the reader to [29] for a more detailed description of the protocol.

4.2 Recovery wallets

Argent [1] is a wallet solution combining the proactive off-chain recovery and the reactive smart-contract guardian recovery methods discussed in Section 2.2. Argent encrypts the private key with a KEK stored on Argent's servers, while the encrypted private key is stored on the user's cloud storage. As an alternative recovery method, Argent provides a guardian-assisted recovery by enabling the user-designated guardians to rotate the user's keys after a 2FA authentication and a 48-hour time delay, which is still reconfigurable. Note that the guardian method is still reactive, as the wallet has pre-selected the Argent organization as a default guardian, while the user can still add more guardians (with the recovery request requiring a signature of the majority of them), or even remove Argent entirely from the default guardian role. The guardian set however still remains publicly visible, which can have privacy implications for the user.

Braavos [3] is a StarkNet account abstraction smart contract wallet that manages 2 keys: one key is stored by the user's device HSM, while the other is deterministically derived from a seed phrase. A 2-out-of-2 multisig is required to execute a transaction. In case the device key is lost, a user initiates a recovery request using the seed phrase through a "Remove Hardware Signer Request" transaction. There is a (configurable) delay for this request to be completed, enabling the user to transfer the funds out of the wallet in case the seed is compromised. **ZenGo** [21] is a wallet that manages a secret-shared key. The first share (Personal Share) is created and stored on the user's device, while the second share (Remote Share) is created and secured on ZenGo's servers. To execute a transaction, both secret shares are used to sign. ZenGo offers a proactive backup recovery method by encrypting the personal share in the user's cloud service, using a KEK which is stored at ZenGo servers. To initiate the recovery, the user needs to provide biometric information to ZenGo (e.g., face recognition). To enable recovery of the remote share, the user also keeps the remote share encrypted with ZenGo's "master key". In case ZenGo ceases to operate, a designated law firm will publish the master key to enable recovery of the remote share.

Sequence [17] is a wallet offering 2 different versions. The first uses 2 keys, a Session key and a backup key which is only shown during account creation. The second version has 3 keys: a session key stored on the user's device, a guard key stored in the wallet provider's servers, and a "Torus" key stored at Torus network [18] in a 2-of-3 secret-shared format, shared between the user's device, a secondary device and the Torus network nodes. A standard transaction execution requires 2-of-3 or 1-of-2 multi-sig (depending on the version), while a recovery transaction request can be signed by 2 keys replacing the third one, or using the backup key. The issues with Sequence are that a "Lazy" user can still hold the device keys on a single device anyway, creating a single point of failure, while the backup key has too much power over all other keys as it can be used to sign any transaction by itself.

Coinbase [47] proposes a proactive distributed key solution for recovery through an MPC wallet. This wallet consists of two distinct backup and recovery solutions: The first consists of 2 shares, one held by the user which can be proactively backed up directly to the user's cloud storage, and the second is held by a designated service provider which can be used to restore through strong authentication. The second is by encrypting both shares using a public key belonging to the user, which gives the user the ability to recover the key without relying on an external service provider or mnemonics.

BitGo [2] offers a WaaS product line which offers both custodial and "semicustodial" solutions. The semi-custodial versions come in two flavors, one using multi-signatures and the other using threshold signatures. In both cases, the user generates two keys, a main key, and a backup key, while BitGo holds a key of its own. Although BitGo does not have any access to either of the user's keys, it needs to sign every user-generated transaction using its own key. To perform recovery in case of key loss, the user needs to store a "keycard" which (among others) has the two keys in an encrypted format. Then the user using the encrypted keys combined with a user passphrase and a number of other inputs, builds a transaction that sends the account funds to a new address [13].

4.3 Systematization

In Table 1 we categorize our findings based on the methodology discussed in Section 3, while in Table 2 we focus on the tradeoffs within the usability vs.

Solution	Action	Recovery type	Account type	Guardian	Multi-key support	Custody	
EIP2429	Reactive	Funds	SC	Yes	No	Self-custodial	
ERC4337	Reactive	Funds	SC	Yes	No	Self-custodial	
KELP	Reactive	Funds	SC	No	No	Self-custodial	
Argent	Both	Both	SC	Yes	No	Self-custodial	
Braavos	Proactive	Account	SC	No	Yes	Self-custodial	
Sequence	Proactive	Account	Standard	No	Yes	Self-custodial	
ZenGo	Proactive	Account	Standard	No	Yes	Non-custodial	
Coinbase	Proactive	Account	Standard	No	Yes	Non-custodial	
BitGo	Proactive	Funds	Both	No	Yes	Non-custodial	

Table 1: Systematization on existing recovery methods. Here, SC means Smart Contract

Solution	EIP2429	ERC4337	KELP	Argent	Braavos	Sequence	ZenGo	Coinbase	BitGo
Usability	 ✓ 	 ✓ 	~	×	×	×	×	 ✓ 	×
Availability	 ✓ 	 ✓ 	~	 ✓ 	~	×	×	 ✓ 	×
Malicious recovery	~	~	×	×	~	~	~	~	✓
Recovery denial	×	×	~	~	~	~	×	×	>
Key com- promise	~	~	~	~	~	×	×	×	×
Privacy	 ✓ 	×	×	×	×	×	/	 ✓ 	/

Table 2: Recovery solution properties.

availability vs. security trilemma, where we distill the security aspects based on the attack vectors discussed. Also recall that purely reactive solutions by definition offer the best usability.

5 Research insights and gaps

With our systematization results in place, we now provide our overall observations, followed by a number of interesting research directions in the Web3 recovery space. We first discuss the research insights we have gathered as a part of this work. We then present some gaps that are evident from our analysis.

5.1 Research Insights

First, we observe that solutions that use a smart contract account are typically reactive and can only recover the funds but not the account themselves. On the other hand, in standard accounts, the recovery is typically performed for the whole account, but the method is proactive. **Insight 1** There is a high correlation between reactive recovery, fund recovery, and smart contract accounts. Similarly, there is a high correlation between proactive recovery, account recovery, and standard accounts.

As discussed in Section 2.2, low-entropy mnemonics such as PINs and passwords cannot be used for recovery purposes when the key is lost. Therefore, we consider such mnemonics as Web2 solutions. However, such mnemonics can be used to encrypt MPC keys, which can in turn perform recovery for a missing key in an MPC wallet.

Insight 2 While low-entropy mnemonics cannot be used by themselves for recovering a lost key, they can be a useful tool for encrypting key shares in MPC wallets, which in turn can perform recovery of lost MPC keys.

Then, a natural question is if managing shares provides more availability than managing high-entropy mnemonics (e.g., 12 random words). Indeed, key shares can be distributed in digital format across many places (e.g., cloud storage, offline media etc.), while it is harder to store mnemonics in more than one place, which makes it less available.

Insight 3 Key shares provide higher availability for recovery purposes compared to high-entropy mnemonics.

Finally, smart contract-based recovery systems seem to offer good design solutions when combined with MPC wallets for transferring funds to heirs after the account holder passes away. Such designs can involve a key that is not sufficient to spend but can still invoke a recovery.

Insight 4 A weak "dead man's key" as part of an MPC architecture within a smart contract-based recovery can provide a method to transfer funds to the heirs.

5.2 Gaps

When considering a recovery method that involves backup keys or MPC wallets, the general recommendation is to transfer the account assets to a new account using the remaining key(s). However, this recommendation is problematic when a key belonging to an important entity in the blockchain ecosystem (e.g. a validator or a guardian) is lost, as this opens up the possibility of other attacks not applicable to a simple account (e.g. a long range attack for validators).

Gap 1 When keys are lost by entities in a blockchain system that have special powers such as validators or guardians, the recovery methods available for users might not be recommended or applicable. Key recovery systems in blockchains need to be carefully designed to address such cases.

We have examined several wallet proposals that integrate multiple recovery methods into a single solution. For instance, Argent [1] merges off-chain recovery with a guardian-based approach to enhance user accessibility. However, while these methods may be secure individually, their combination could potentially create a vulnerable system. For instance, a backup key could be misused to execute an unauthorized guardian recovery. Therefore, while the intention of combining recovery methods is to increase security and availability, it is crucial to carefully consider potential security risks that might arise from such combinations.

Gap 2 Combining different recovery schemes can negatively impact the overall security of a web3 system.

We also observe from Tables 1 and 2 that while on-chain recovery methods are generally reactive (which is a desired property), none of them preserve users' privacy, as they expose to the public that the account holder has made a recovery request. Also, no reactive solution exists for blockchain systems that offer privacy-preserving characteristics to the user, e.g., confidentiality or anonymity.

- **Gap 3** a. Can we design reactive recovery methods that do not reveal that a recovery is in process?
- b. Can we design reactive recovery methods for privacy-preserving blockchains?

We also observe that all (desirable) reactive methods require a smart contract account. However, there are no reactive methods available for standard accounts.

Gap 4 Can we design a reactive recovery solution for standard accounts?

In the introduction, we discussed that mistyping the recipient's address is effectively equivalent to key loss, and such cases are usually mitigated through checksums or QR codes. However, these mitigations can be cumbersome and introduce an overhead for the user and the application.

Gap 5 Can mistyped addresses be prevented by automated logic (e.g. a smart contract) which would check if an address actually exists before transferring funds?

Additionally, we discussed that account recovery methods are applicable for keys that are in a "lost" state, but are not applicable for keys that are in a "leaked" or "stolen" state (with only a few special exceptions which involve key rotation, and assume that the attacker is not actively attempting to make a competing recovery attempt).

Gap 6 Can we design a secure and available account recovery scheme that works for leaked keys?

A potential research area would be exploring novel smart contract (plus notification) systems that introduce transaction execution delays on purpose (i.e., via time-lock [57] and/or fraud proof [24] logic) so that a user whose key is compromised can monitor unintended transactions and stop them from finalization. It is still an open problem though how such a system would prevent vicious cycle scenarios, where the account owner tries to stop malicious transactions, while the adversary prevents the legitimate user from transferring their assets. Another interesting avenue is linking identity (with privacy) to blockchain address derivation, such as in zkLogin [35], and then recovering via identity proofs. **Gap 7** Standardization of applying a user-experience friendly time-delayed transaction system or identity proofs to prevent malicious activity in the case of compromised keys is an interesting, open problem.

Finally, we described how modern wallets apply secret sharing methods by splitting a key between multiple entities and storage mediums. The reality is that most of the storage options for the shares can be weaker (in terms of security guarantees) compared to the recommendations for storing cryptographic keys. For instance, many threshold wallets store key shares in Google Drive or their back-end premises. However, if the key (mnemonic) was not split and stored as a whole, the official security recommendations will not necessarily be using systems like cloud file-systems or a startup's database, but preferring password managers, hardware secure modules or offline vaults that have been designed for that purpose. Additionally, an attack to any of these key-share storage mediums would reveal one of the shares for all users using that system, which would imply that the threshold recovery property is lost and all accounts will be downgraded from i.e., a 2-of-3 to an 1-of-2 structure, and thus thousands of millions of accounts will get affected, resulting in initiating the recovery process (and transfer of assets to a new account) en masse.

Gap 8 Community is lacking a rigorous and quantitative analysis around storing multiple shares in independent, but potentially centralized and weaker storage mediums versus sacrificing recoverability but storing them in strong key-storage recommended solutions.

6 Related Work

In this section we mention some wallets that have lesser recovery functionalities, therefore we do not include them is our systematization. We also review related work that is more broadly related to recovery in a blockchain setting.

6.1 Other Relevant Cryptocurrency Wallets

Circle [4] offers a WaaS solution known as "Programmable Wallets". In their non-custodial version, referred to as "User Controlled", users establish a PIN code and answer security questions during the initialization process. This setup allows users to restore their PIN if they ever forget it. In this wallet, the user's private key is stored on their phone, and the PIN serves as the encryption key for the private key.

The recovery process in this wallet pertains to restoring the PIN, which can only be accomplished by correctly answering the security questions set during the setup. However, the user cannot restore the PIN independently. Before answering the security questions, the user needs to connect to Circle's servers to obtain an encryption key and a challenge ID to carry out the restoration process.

Consequently, the user is not entirely self-reliant in restoring the PIN, and if the user's device is lost, recovery of the wallet's assets becomes impossible. This highlights the importance of safeguarding the device where the wallet is installed and remembering the answers to the security questions.

Safe [16] follows a modular design for creating smart contract accounts, where add-on "modules" can be used to add functionalities or features to the core contract. Among the modules proposed is a "recovery" module which uses guardians to recover the funds associated with the account when the private key is lost.

Magic [11] provides a WaaS solution similar to Circle, and the recovery functionality is limited to the user's email which is used as a method to login to the wallet. In case the user loses access to his/her email, Magic provides an "account recovery" feature, where a secondary method of authentication using phone SMS has to be proactively enabled. As in Circle, Magic does not offer an explicit account recovery functionality in case the phone is lost.

6.2 Other Related Work

Firstly, a number of works performed taxonomies of wallets [43,55,38], however these works focus primarily aspects such as security and performance and not recovery. Still some of those wallets proposed recovering the account using a seed phrase or even splitting the private key into encrypted secret shares in a distributed network [41]. We consider those as the first attempts in the account recovery efforts in a blockchain setting.

In 2020, a special decentralized autonomous organization (DAO) was proposed as part of the Ethereum's ecosystem [10]. That DAO would enable users to recover their funds in the case of mistakenly sending those to a wrong, non-existent address (e.g. by mistyping one character). The core idea behind this proposal was that the DAO would transfer donated funds to addresses with only one or two characters difference.

Abramova et al. [23] performed a study of 359 crypto-asset users and grouped them into three categories cipherphunks, holders, and rookies based on their perceived notions of security and self-efficacy in protecting their keys. Each of these groups has different concerns regarding the security and recovery of their keys. In a recent study [49], the authors investigate user perceptions and preferences concerning multi-device cryptocurrency wallets, aiming to bridge the gap between the wallets' designed security models and users' mental models. The study involves 357 participants and employs both qualitative and quantitative analyses to gain insights into current wallet usage patterns, choices, and the willingness to transition from single-device to multi-device wallets. The findings reveal several results: (1) The majority of participants primarily use single-device wallets, yet over 80% express concerns about potential fund loss or key compromise. (2) When provided with essential knowledge about both single and multi-device wallets, 54.7% of participants express their readiness to shift to multi-device wallets. (3) Regarding key-management and architectural preferences, 60.8% of participants prefer a small number of reputable servers, while 34.6% choose a higher number of servers. Their preferences were also significantly affected by the key shares are hosted. This shows that the users choice of keys regarding security (and by extension recovery) of the keys is affected by the government policy and the reputation of the servers hosting the key shares.

In a related line of research the usable security and privacy researchers explored the social authentication paradigm for authentication and account recovery [44,52,51,42,25]. In social authentication, account holders rely on their trustees to help them authenticate. Schechter et al. [52] envision such social authentication as a backup option to regain access to the account. Alomar et al. surveyed the existing social authentication schemes and identified that they can be broadly classified into two types— knowledge-based and trust-based [25]. Facebook deployed knowledge-based social authentication where they asked the users to identify photos of their friends—this scheme is broken by Kim et al, and Polakis et al. using facial recognition [44,51]. In response Jain et al., proposed knowledge-based social authentication using more private information about users' interaction with social contacts [42]. On the other hand, trust-based social authentication employs offline communication with trustees of a user for authentication [25]. These trustees can be chosen by user, by system, or by network properties. We build on this line of work—however, our context is quite different than these previous efforts. Instead of considering social-relationshipaware systems (e.g., Facebook) that most of these previous works considered, we particularly considered blockchain recovery, where the onus is on the user to ensure the recovery mechanism to recover their account (or keys). Our work sheds light on the unique usability-security trade-off for users while setting up social authentication in this context.

Also, in some proposals there is no private key management for the user at all. For instance, "keyless" wallets [59] use one-time passwords to enable spending funds, while in Qredo [15] the key is held by servers in a "decentralized MPC" fashion and the user is authenticated with biometrics. Because such proposals deviate from the classic private key management approach and a "key loss/recovery" is not applicable, we consider those as out of scope in our systematization work. In addition, we do not treat key rotation techniques used proactively when a key compromise is suspected as account recovery. Finally, in custodial wallets where the user does not need manage any private key, but delegates the account management to an external service provider who holds the key on the user's behalf, an account recovery is essentially a web2 recovery where a user will need to perform functionalities such as a password reset. Therefore, we also consider recovery in custodial wallets as out of scope. However, given the ambiguity of the "custody" aspect in the Web3 space, we provide definitions of the various wallet types in Section 3.

7 Conclusion

While account recovery is a well-established concept in the context of Web2, it presents unique challenges in the Web3 landscape. In Web3, accounts rely on a secure signing-verification key pair, which is not shared with a central entity with Web2 focus on security via decentralization. This distinction makes Web3 account recovery more complex. Fortunately, the Web3 community is experiencing increased interest and effort into recovery mechanisms to provide users confidence that their assets will be available even if they do not have the expertise on managing cryptographic keys. While we have observed innovative designs that attempt to bridge this gap, the existing solutions are imperfect, and do not strike a balance between usability, security and availability.

This study systematically investigates existing account recovery solutions in the blockchain realm, analyzing their workflows, cryptographic techniques, and unique characteristics. We shed light on a trilemma involving usability, security, and availability that arises in Web3 recovery scenarios as a natural extension of security and usability dilemma in Web2 and security and availability tradeoff in distributed systems. By categorizing existing recovery mechanisms along different axes that highlight trade-offs, we offer valuable insights and suggest future research directions for improving Web3 account recovery.

References

- Argent vault security and recovery, https://support.argent.xyz/hc/enus/articles/360007338877-How-to-recover-my-wallet-with-guardianscomplete-guide
- 2. Bitgo docs, https://developers.bitgo.com/guides/wallets/recover
- Braavos, https://braavos.notion.site/Hardware-Protected-Signer-FAQs-5d5ae07e999e45ddaf8a7f5c4abbad80
- Circle's web3 services overview, https://developers.circle.com/w3s/docs/ user-controlled-recover-account
- Coindesk: Custodial vs non-custodial wallets, https://tinyurl.com/coindeskcustodial-wallet
- Crypto.com:custodial vs non-custodial wallets, https://tinyurl.com/cryptocustodial-wallet
- Custodial vs. non-custodial wallets, https://www.gemini.com/cryptopedia/ crypto-wallets-custodial-vs-noncustodial#section-non-custodialcrypto-wallets-taking-personal-responsibility-for-your-assets
- Custodial vs non-custodial wallets, https://crypto.com/university/custodialvs-non-custodial-wallets
- Custodial wallets vs. non-custodial crypto wallets, https://www.coindesk.com/ learn/custodial-wallets-vs-non-custodial-crypto-wallets/
- 10. Eth community discuss dao for reversing funds lost to wrong addresses, https://cointelegraph.com/news/eth-community-discuss-dao-forreversing-funds-lost-to-wrong-addresses
- Magic developer sdk, https://magic.link/docs/dedicated/enterprise/ account-recovery#enabling-account-recovery
- 12. A man who says he threw away a hard drive loaded with 7,500 bitcoins in 2013 is offering his city \$70 million to dig it up from the dump, https://www.businessinsider.com/man-offers-council-70-milliondig-up-bitcoin-hard-drive-2021-1
- 13. Non bitgo recovery instructions, https://github.com/BitGo/wallet-recoverywizard/blob/master/NON_BITGO_RECOVERY.md

- 14. People are losing bitcoin cash by accidentally sending it to bitcoin addresses, https://www.vice.com/en/article/j55ywg/people-are-losing-bitcoincash-by-accidentally-sending-it-to-bitcoin-addresses
- 15. Qredo whitepaper, https://www.qredo.com/qredo-white-paper.pdf
- 16. Safe core protocol overview, https://docs.safe.global/safe-core-protocol/ safe-core-protocol
- 17. Sequence wallet key management, https://docs.sequence.xyz/key-management
- 18. Torus network documentation, https://docs.tor.us/
- What happens to cryptocurrency when you die?, https://fortune.com/crypto/ 2017/09/26/cryptocurrency-bitcoin-death/
- 20. What's a self-custodial wallet?, https://www.bitcoin.com/get-started/ custodial-non-custodial-bitcoin-wallets/#whats-the-differencebetween-selfcustodial-and-noncustodial
- Zengo wallet recovery, https://zengo.com/understand-the-difference-zengowallet-recovery-vs-ledger-recover/
- 22. The aptos blockchain: Safe, scalable, and upgradeable web3 infrastructure (August 2022), https://aptos.dev/assets/files/Aptos-Whitepaper-47099b4b907b432f81fcOeffd34f3b6a.pdf
- Abramova, S., Voskobojnikov, A., Beznosov, K., Böhme, R.: Bits under the mattress: Understanding different risk perceptions and security behaviors of cryptoasset users. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. CHI '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3411764.3445679, https://doi.org/ 10.1145/3411764.3445679
- Al-Bassam, M., Sonnino, A., Buterin, V.: Fraud proofs: Maximising light client security and scaling blockchains with dishonest majorities. arXiv preprint arXiv:1809.09044 160 (2018)
- Alomar, N., Alsaleh, M., Alarifi, A.: Social authentication applications, attacks, defense strategies and future research directions: A systematic review. IEEE Communications Surveys & Tutorials 19(2), 1080–1111 (2017)
- Amsden, Z., Arora, R., Bano, S., Baudet, M., Blackshear, S., Bothra, A., Cabrera, G., Catalini, C., Chalkias, K., Cheng, E., Ching, A., Chursin, A., Danezis, G., Giacomo, G.D., Dill, D.L., Ding, H., Doudchenko, N., Gao, V., Gao, Z., Garillot, F., Gorven, M., Hayes, P., Hou, J.M., Hu, Y., Hurley, K., Lewi, K., Li, C., Li, Z., Malkhi, D., Margulis, S., Maurer, B., Mohassel, P., de Naurois, L., Nikolaenko, V., Nowacki, T., Orlov, O., Perelman, D., Pott, A., Proctor, B., Qadeer, S., Rain, Russi, D., Schwab, B., Sezer, S., Sonnino, A., Venter, H., Wei, L., Wernerfelt, N., Williams, B., Wu, Q., Yan, X., Zakian, T., Zhou, R.: The libra blockchain (2020), https://diem-developers-components.netlify.app/papers/the-diemblockchain/2020-05-26.pdf
- Arora, S.S., Badrinarayanan, S., Raghuraman, S., Shirvanian, M., Wagner, K., Watson, G.: Avoiding lock outs: Proactive FIDO account recovery using managerless group signatures. Cryptology ePrint Archive, Report 2022/1555 (2022), https://eprint.iacr.org/2022/1555
- 28. Biddle, R., Chiasson, S., Van Oorschot, P.C.: Graphical passwords: Learning from the first twelve years. ACM Computing Surveys (CSUR) 44(4), 1–41 (2012)
- Blackshear, S., Chalkias, K., Chatzigiannis, P., Faizullabhoy, R., Khaburzaniya, I., Kogias, E.K., Lind, J., Wong, D., Zakian, T.: Reactive key-loss protection in blockchains. Cryptology ePrint Archive, Report 2021/289 (2021), https:// eprint.iacr.org/2021/289

- Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: International Workshop on Public Key Cryptography. pp. 31–46. Springer (2002)
- Buterin, V., Van de Sande, A.: EIP-55: Mixed-case checksum address encoding (2016), https://eips.ethereum.org/EIPS/eip-55
- 32. Buterin, V., Weiss, Y., Tirosh, D., Nacson, S., Forshtat, A., Gazso, K., Hess, T.: Erc-4337: Account abstraction using alt mempool (2021), https://eips. ethereum.org/EIPS/eip-4337
- Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: 28th ACM STOC. pp. 639–648. ACM Press (May 1996). https://doi.org/10.1145/237814.238015
- Chalkias, K.: Soft privacy-related leak in some threshold edds wallets (2023), https://x.com/kostascrypto/status/1703594584100700641?s=20
- 35. Chalkias, K., Ji, Y., Lindstrøm, J., Maram, D., Riva, B., Roy, A., Sedaghat, M., Wang, J.: Zk for authentication: How to snark sign-in w/ google, apple & facebook. zkSummit 10 (2023), https://docs.google.com/presentation/d/ 1XBQ8d7BorBZocG8uvmnJkQN6Apvdsw7Ntfd3uJuqxII
- Dawson, E., Donovan, D.M.: The breadth of shamir's secret-sharing scheme. Comput. Secur. 13(1), 69–78 (1994)
- Duggan, G.B., Johnson, H., Grawemeyer, B.: Rational security: Modelling everyday password use. International journal of human-computer studies **70**(6), 415–431 (2012)
- Erinle, Y., Kethepalli, Y., Feng, Y., Xu, J.: Sok: Design, vulnerabilities and defense of cryptocurrency wallets. CoRR abs/2307.12874 (2023). https://doi.org/10.48550/arXiv.2307.12874, https://doi.org/10.48550/arXiv. 2307.12874
- Gaw, S., Felten, E.W.: Password management strategies for online accounts. In: Proceedings of the second symposium on Usable privacy and security. pp. 44–55 (2006)
- Harn, L.: Group-oriented (t, n) threshold digital signature scheme and digital multisignature. IEE Proceedings-Computers and Digital Techniques 141(5), 307–313 (1994)
- He, X., Lin, J., Li, K., Chen, X.: A novel cryptocurrency wallet management scheme based on decentralized multi-constrained derangement. IEEE Access 7, 185250-185263 (2019). https://doi.org/10.1109/ACCESS.2019.2961183, https:// doi.org/10.1109/ACCESS.2019.2961183
- 42. Jain, S., Gong, N.Z., Basuroy, S., Lang, J., Song, D., Mittal, P.: New Directions in Social Authentication. In: Proceedings of the Workshop on Useable Security (USEC'15). San Diego, CA (February 2015)
- 43. Karantias, K.: Sok: A taxonomy of cryptocurrency wallets. Cryptology ePrint Archive, Paper 2020/868 (2020), https://eprint.iacr.org/2020/868, https: //eprint.iacr.org/2020/868
- 44. Kim, H., Tang, J., Anderson, R.: Social authentication: Harder than it looks. In: Keromytis, A.D. (ed.) Financial Cryptography and Data Security. pp. 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
- Lewis, J.R., Utesch, B.S., Maher, D.E.: Umux-lite: when there's no time for the sus. In: Proceedings of the SIGCHI conference on human factors in computing systems. pp. 2099–2102 (2013)
- Lewis, J.R., Utesch, B.S., Maher, D.E.: Investigating the correspondence between umux-lite and sus scores. In: Marcus, A. (ed.) Design, User Experience, and Usability: Design Discourse. pp. 204–211 (2015)

- Lindell, Y.: Cryptography and mpc in the coinbase prime web3 wallet (2023), https://coinbase.bynder.com/m/68b5c8bedf49a9f9/original/CB-Prime-W3W-MPC-White-Paper.pdf
- Makriyannis, N., Yomtov, O.: Practical key-extraction attacks in leading mpc wallets. Cryptology ePrint Archive, Paper 2023/1234 (2023), https://eprint.iacr. org/2023/1234, https://eprint.iacr.org/2023/1234
- 49. Mangipudi, E.V., Desai, U., Minaei, M., Mondal, M., Kate, A.: Uncovering impact of mental models towards adoption of multi-device crypto-wallets. Cryptology ePrint Archive, Report 2022/075 (2022), https://eprint.iacr.org/2022/075
- Maram, D., Kelkar, M., Eyal, I.: Interactive authentication. Cryptology ePrint Archive, Report 2022/1682 (2022), https://eprint.iacr.org/2022/1682
- Polakis, I., Lancini, M., Kontaxis, G., Maggi, F., Ioannidis, S., Keromytis, A.D., Zanero, S.: All your face are belong to us: Breaking facebook's social authentication. In: Proceedings of the 28th Annual Computer Security Applications Conference. p. 399–408. ACSAC '12 (2012)
- 52. Schechter, S., Egelman, S., Reeder, R.W.: It's not what you know, but who you know: A social approach to last-resort authentication. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. p. 1983–1992. CHI '09 (2009)
- 53. Schmidt, R., Mota, M., Buterin, V., naxe: Ethereum EIP2492 secret multisig recovery (2019), https://gitlab.com/status-im/docs/EIPs/blob/secretmultisig-recovery/EIPS/eip-2429.md
- Shoup, V.: Practical threshold signatures. In: Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19. pp. 207–220. Springer (2000)
- Suratkar, S., Shirole, M., Bhirud, S.: Cryptocurrency wallet: A review. In: 2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP). pp. 1–7 (2020). https://doi.org/10.1109/ICCCSP49186.2020.9315193
- 56. Swambo, J., Poinsot, A.: Risk framework for bitcoin custody operation with the revault protocol. In: Bernhard, M., Bracciali, A., Gudgeon, L., Haines, T., Klages-Mundt, A., Matsuo, S., Perez, D., Sala, M., Werner, S. (eds.) Financial Cryptography and Data Security. FC 2021 International Workshops. pp. 3–20. Springer Berlin Heidelberg, Berlin, Heidelberg (2021)
- 57. Tsabary, I., Yechieli, M., Manuskin, A., Eyal, I.: Mad-htlc: Because htlc is crazycheap to attack. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 1230–1248 (2021). https://doi.org/10.1109/SP40001.2021.00080
- 58. Yao, A.C.C.: Protocols for secure computations (extended abstract). In: 23rd FOCS. pp. 160–164. IEEE Computer Society Press (Nov 1982). https://doi.org/10.1109/SFCS.1982.38
- Zindros, D.: Hours of horus: Keyless cryptocurrency wallets. In: Matsuo, S., Gudgeon, L., Klages-Mundt, A., Perez Hernandez, D., Werner, S., Haines, T., Essex, A., Bracciali, A., Sala, M. (eds.) Financial Cryptography and Data Security. FC 2022 International Workshops. pp. 615–644. Springer International Publishing, Cham (2023)

A Formal definitions

Secret sharing. A Threshold Secret Sharing scheme consists of two algorithms Share and Reconstruct as described below:

- Share $(\lambda, n, t) \rightarrow \{s_i\}_{i \in [n]}$: It takes the security parameter λ , and the access structure parameters (n, t) as inputs. n is the total number of parties and t is the threshold for reconstruction. It outputs n shares, corresponding to each individual party.
- Reconstruct $(\{s_j\}_{j \in \mathcal{R}}) \to s$: The reconstruction function takes shares from a set \mathcal{R} of valid shares of size at least t + 1 and outputs the reconstructed secret s.

Threshold signature scheme. A threshold signature scheme consists of algorithms to generate key shares, generate partial signatures, aggregate the partial signatures into a threshold signature, and verify any partial or threshold signature.

- keygen $(1^{\lambda}, n, t) \rightarrow \{sk_i, pk_i\}_{i \in [n]}$: The key generation algorithm generates private secret key and verification key shares such that any t + 1 or more of the secret key shares can reconstruct the system secret key sk.
- $\operatorname{parsign}(sk_i, m) \to \sigma_i$: Each signer indexed $i, i \in [n]$ generates a partial signature σ_i using his secret key share sk_i .
- aggregate $\{\sigma_i\}_{i \in \mathcal{R}} \to \sigma$: The signature aggregation algorithm takes any set of t + 1 or more valid partial signatures to produce the threshold signature σ .
- verify $(\sigma, m, pk) \rightarrow 1/0$: The verification algorithm takes the signature (threshold or partial) along with the corresponding verification key. If the verification is successful, it outputs 1, else 0.

Multi-signature scheme. A Multi-signature scheme consists of the following algorithms:

- keygen $(1^{\lambda}) \rightarrow \{sk, vk\}$: The key generation algorithm generates the private secret key and the corresponding verification key. This is run by each party indexed *i* to generate their local key pairs (sk_i, vk_i)
- $\operatorname{sign}(sk, m) \to \sigma$: The signature algorithm takes the secret key and the message *m* to generate a signature σ . Each party indexed *i* generates a signature σ_i using his secret key sk_i .
- verify $(\sigma, m, vk) \rightarrow 1/0$: The verification algorithm takes a signature and the corresponding verification key. If the verification is successful, it outputs 1, else 0.
- muSigVerify($\{\sigma_i, vk_i\}_{i \in \mathcal{R}}, m, n, t\} \rightarrow 1/0$: The verification algorithm takes a set of t + 1 or more signatures and verifies each of them using the verify algorithm. If at least t + 1 of them return 1, the algorithm returns 1, else 0.