mathematics

Article

Resilient Custody of Crypto-Assets, and Threshold Multisignatures

Vincenzo Di Nicola [1], Riccardo Longo [2,*], Federico Mazzone [2] and Gaetano Russo [2]

1 Conio Inc., San Francisco, CA 94105, USA; vincenzo@conio.com
2 Department of Mathematics, University of Trento, Povo, 38123 Trento, Italy; federico.mazzone1@gmail.com (F.M.); rs.gaetano@gmail.com (G.R.)
* Correspondence: riccardolongomath@gmail.com

Received: 4 September 2020; Accepted: 1 October 2020; Published: 14 October 2020

Abstract: Ensuring safe custody of crypto-assets, while at the same time allowing a user to easily access and transfer them, is one of the biggest challenges of this nascent industry. This is even harder considering also the multiple technological implementations behind each crypto-asset. Here we present a survey of the various solutions for this custody problem, comparing advantages and disadvantages. Then we delve deeper into some interesting schemes based on secure multiparty computation, which give a blockchain-agnostic solution that balances security, safety, and transactional ease of use, and in particular, a protocol that enhances practicality by exploiting a party that may stay offline during the key generation.

Keywords: custody of cryptocurrencies; digital assets; applied cryptography; threshold multisignatures; secure multiparty computation

1. Introduction

Custody of crypto-assets, such as cryptocurrencies, is at the very core of the burgeoning digital-asset market. Safeguarding the private key associated with the crypto-asset presents, in fact, many issues, especially for the general public. Private keys can be easily lost or forgotten (see, for example [1]), leading to the inaccessibility of the related assets. Serious issues arise also in case of inheritance following a death: crypto-assets cannot be inherited unless the heir already has access to the private key.

Several major financial institutions around the world are trying to tackle this challenge: some by building internal teams within their organization, such as Fidelity in the US [2]; others by leveraging the expertise of startups, such as Nomura in Japan with Ledger [3] or leading challenger bank Hype in Italy with Conio [4].

Many solutions have been devised to mitigate these problems and to enable safe custody; we will look at some of them in Section 2. Some rely on personal efforts (e.g., cold storage), others simply delegate full control of the assets to a third party. Unfortunately, these kinds of solutions are partial: most either sacrifice easiness and readiness of access to the assets, or completely rely on the trustworthiness of a third party, or are so "secure" that there is a risk of losing access forever.

In recent years a new technological and operational approach has been developed: threshold multisignatures, which we will discuss in Section 3. This kind of solution addresses more comprehensively the problems above. It relies on three private keys, instead of a single one, which are distributed among three parties, and two of them are required to transfer the crypto-assets. This approach is resilient with respect to the unavailability of one party. For instance, if the user ends up losing the private key, the other two parties can transfer the assets to a new wallet. Security is guaranteed as long as the two helping parties do not collude, i.e., it is sufficient that one of the two remains honest to preserve the safety of the system. Furthermore, this solution is effectively agnostic to the underlying blockchain, i.e., it does not have to be supported by special features.

2. Traditional Custody of Crypto-Assets

Any solution to the custody problem has to face the trade-off between security and spendability, where the latter means that signing a transaction should be fast and easy. The perfect solution should balance the two extremes and, in particular, satisfy the following desirable properties:

1. *ease of use*: a legitimate user should be able to access the funds without hassle;
2. *security*: adversaries should not be able to gain control over the assets;
3. *safety*: the system should maintain or recover accessibility to the funds in case of accidents (e.g., loss of the device normally used);
4. *universality*: the solution should be applicable to all crypto-assets and not rely on some special feature.

Now, we will discuss the traditional solutions to the problem that are currently adopted, starting from the most trivial ones and proceeding towards the state of the art, showing their features and analyzing some pros and cons in various use cases.

A perfect classification of these methods is not possible: some of them are not even comparable with each other as they solve different parts of the problem, while others may be combined and used together. Thus, we do not aim to create an exhaustive list of all existing solutions to the problem, but instead provide an overview of the solutions currently used in practice.

2.1. Self-Custody

One of the most immediate and trivial solutions to the problem of crypto-assets custody is self-custody: no third party is involved—the user is entirely responsible for the security and protection of the private key. An aid may come from a mnemonic passphrase (usually comprised of 12 or 24 common words) that can be used to reconstruct the key and, for example, facilitate the transfer to a new device. This solution is particularly appreciated by those who want complete control of their digital assets, usually due to a lack of trust or confidence in a third party's transparency, policies, and behavior.

In more concrete terms, users can implement self-custody by saving their private key somewhere, creating their own custom storage and backup systems. For example, the key may be kept on a private device (such as a smartphone, tablet, or laptop), thus giving the user easy access to the relative funds, yet, on the other hand, increasing the risk of theft or hacking.

Another solution could be to use hardware wallets that keep the key offline and require a PIN to unlock it. Here, however, we must take into account the risks related to device loss and hardware failure, as well as the problem of memorizing and/or safely storing the PIN. These solutions will be better analyzed in Section 2.3. A low-tech version of these wallets is a printed version of the private key, but of course the risk of losing or damaging a piece of paper is even greater.

Finally, the user could opt to combine these methods, saving multiple copies of the private key on different media. While this reduces the risk of loss, improving safety, it also increases the risk of theft, decreasing the security.

In conclusion, great responsibilities come with great risks: the mismanagement of your private key could mean losing access to your funds forever.

2.2. Third-Party Custody

Another solution for custody of crypto-assets involves a third party that holds the private key and manages the user's funds. The user is able to access the wallet through standard authentication methods managed by the third party, usually a "traditional" password and perhaps a two-factor authentication. In this way, the security is completely in the hands of this third party, which must manage both the secure storage of private keys and the security of user accounts. The external entity acts following the instructions of the user, who is never directly involved in the signature process.

As previously mentioned, some users tend to not trust these services, often due to the lack of transparency about security policies.

The main examples of this kind of third-party custodians are exchange platforms, which, in addition to taking care of the user's funds, also offer the possibility to buy and sell cryptocurrencies. These kinds of custody services are called "hot" wallets since they are constantly connected to the Internet. Some well-known exchanges are Coinbase [5], Kraken [6], and Bitstamp [7]; they allow the storing and exchanging of a large number of different cryptocurrencies.

The advantages of totally relying on a third-party service are quite obvious: the user offloads the responsibility of the protection of the private key from attackers and from the possibility of losing it. On the other hand, the disadvantage, as already mentioned in Section 2.1, is the absence of direct control over the private key: the user has to trust that the service provider is adequately secure, that it is always available to carry out transactions, that it respects privacy, and that it actually has the funds it declares having. Historically, there have been multiple examples where security and availability have not been up to expectations—for example, Bitfinex, a Bitcoin exchange that suffered a DDoS attack that interrupted the service for 84 minutes [8]. But the most striking case has been Mt. Gox, a Japanese cryptocurrency exchange that in 2014 handled over 70% of global Bitcoin transactions. In the same year they declared bankruptcy, caused by the loss and theft of over 850,000 bitcoins over the previous four years, due to the mismanagement of the storing system [9].

2.3. Cold Storage

Cold storage, as mentioned in Section 2.1, is a method to improve the security of keys that is widely used among retail customers. This approach relies on electronic and non-electronic devices that hold the signing key and are physically isolated (or "air-gapped") from the network and, therefore, theoretically unassailable by remote adversaries. The name is in opposition to "hot storage" solutions, mentioned in Section 2.2, which are by definition always connected to the Internet.

Cold storage is a transversal solution with respect to the self-custody versus third-party custody paradigm. For the sake of simplicity, we can think of it as a form of self-custody, since the responsibility for keeping the physical medium safe lies completely with the user. However, many online services use a combination of hot and cold storage to enhance security while maintaining high availability, so the adoption of cold-storage solutions is not limited to private users.

The advantages of this type of solution are quite evident: isolating the private key from the network protects it from hacking attacks. However, there are also several disadvantages: using cold-stored keys to sign a transaction is not as fast and convenient as using hot-stored ones; moreover, the physical medium is subject to risks: it could suffer physical faults, be stolen, or have internal software problems.

There are various types of cold storage:

- *Hardware wallets*. These are specialized offline devices, like a USB drive or a smartcard, that safeguard private keys offline (e.g., Ledger USB Wallet [10], TREZOR [11], and KeepKey [12]). The user must enter a PIN in order to unlock the device and use the private key. The main risks are associated with the device being lost or stolen and to malfunctions that make the crypto-assets inaccessible. Considering the inheritance case, if no prior arrangements with the heirs were made (i.e., sharing PIN and location of the USB device), then the assets cannot be recovered. The relevance of this scenario is demonstrated by the death of Gerald Cotten, CEO and co-founder of Quadriga, a Canadian cryptocurrency exchange, which caused the loss of about 145 million dollars in cryptocurrencies, as they were stored in cold wallets and Cotten was the only one who knew the passphrases and recovery keys [13]. Another example given by the personal experience of an ex-director of the magazine Wired [1] shows how users may simply forget their

PIN, resulting in the impossibility of accessing the crypto-funds, and that vulnerabilities in the hardware can undermine the security.

- *Paper wallets*. This solution, which can be considered a sub-type of hardware wallets, consists in generating the public address and the corresponding private key for a certain ledger and printing them on a sheet of paper, usually also as a QR code for practical purposes. The most dangerous moment in terms of security is the generation of the private key: the generator services who provide the paper wallets usually also provide a software that allows generating the keys and the address offline to thwart online attacks, but of course malware may already be present on the device. Paper wallets are invulnerable to online attacks and hardware or software failures (excluding the registration phase), but they are susceptible to the fragility of the medium in question: tearing, burning, wetting, forgetting, or losing the paper sheet leads to the loss of funds. In order to better cope with the medium's durability, there are kits to engrave the wallet's seed in titanium or steel plates, greatly increasing its safety. Recently, interest in this type of solutions has rekindled, for example with the work [14] of Charles Hoskinson, CEO of IOHK and founder of the Cardano blockchain (ADA), who has devised a complex system in which the paper wallet features the private key in encrypted form and is unlocked using a YubiKey [15], a hardware device used for multi-factor authentication.
- *Offline software wallets*. The two solutions above may not be considered by some people as real cold storage as they do not protect the private key when the signature is generated. In fact, even by properly storing the private key, in the very moment it is transferred (e.g., by scanning the QR code of a paper wallet) from the isolated device to software on an online device, there is the risk of an attack. Offline software wallets solve this problem by signing directly offline and releasing only the signature online, while the private key never leaves the protected device in any way. In general, the operation is quite simple: an unsigned transaction is prepared online with all the necessary data (public addresses, amount, . . .), then the transaction is passed on to the offline device with the signing software through a secure medium, then transaction is signed offline and returned. These kinds of solutions, such as the ones offered by Armory [16] and Electrum [17], often come in the form of desktop wallets, and they require at least one computer to always be offline.

2.4. Key Recovery

As widely noted so far, one of the major problems that arises in blockchain systems is the recovery of the private key in case of accidents. Unless you completely rely on third-party custody, thus directly eliminating the responsibility of safeguarding the key, the problem of recovering the private key when it is lost is a major issue.

Another traditional and "analog" approach that builds on self-custody solutions to solve the key recovery problem is to rely on a trusted third party, like a public notary, that safeguards a sealed copy of the user's private key. This naturally allows one to safely retrieve control of assets after key loss and manage inheritance in case of death, at the cost of trusting the notary. However, this approach requires active intervention of the trusted third party every time a new key needs to be protected, and the fees associated with this safeguarding service could be quite significant.

Many interesting solutions have been developed in recent years. In [18], the authors propose the use of biometrics such as fingerprints to restore the key: a user's private key is encrypted using a symmetric key derived from the biometric feature and split into n pieces according to Shamir's Secret Sharing scheme [19], each piece is then distributed to trusted nodes. When the user needs to recover the key, k out of n pieces are retrieved and used to recover the encrypted private key, which can be decrypted with the biometric. Even though encrypting and splitting the private key could be a good workaround that gives good resiliency, there is still a *single point of failure* in the encryption of the key. Moreover, the authors' choice of using fingerprints to secure the key is debatable, since people leave their fingerprints practically everywhere and it is also relatively easy to reconstruct them from high-resolution images, which nowadays are far too easy to acquire. Therefore, fingerprints are not a secure replacement for a password, and a more correct approach would be to use biometrics to integrate an authentication system [20].

2.5. On-Chain Multisignature

All of the solutions seen so far for the crypto-assets custody problem can be somehow categorized into *user-responsible* solutions or *third-party-responsible* solutions. An alternative approach to the crypto-asset custody problem is to split up the responsibility of the crypto-assets among more parties in order to avoid a single point of failure. For instance, the Bitcoin [21] protocol natively makes use of the *multisignature* [22], a type of signature that allows a group of users to authorize transactions combining multiple unique key signatures. The multisignature transactions are also referred to as M-of-N transactions, in the sense that N private keys are associated with a Bitcoin address. In order to make a transaction with this address, at least M (the threshold) signatures corresponding to M distinct private keys (out of the N associated with the address) must be affixed.

Dividing the responsibility between different parties increases security since even if some of the private keys are compromised, as long as they are less than M, the address remains secure. It also works well for the key recovery issue: if $n \leq N - M$ users lose their private keys, the remaining users associated with that address still have access to the funds and can move them to a new address.

In addition to providing security and key recovery, this solution has many side applications. For example, it can be used to manage the funds of a large company where private keys are given to N executives and it is desired that the funds can be spent only if at least M of them agree, and the relative importance of the executives can be implemented by giving more keys to some of them. Note that it is not necessary that the various private keys go to separate parties. For instance, using it in the trivial case of N-of-N, it turns into a simple N-factor authentication wallet. By storing the associated private keys on various devices of the same user, an improvement in the self-custody is obtained, at the cost of slightly reducing ease of use and spendability.

Unfortunately, this solution also has several disadvantages. First of all, operating this kind of signature has a cost in terms of transaction fees because the information of the signers has to be included. Furthermore, privacy is slightly reduced since these kinds of addresses are different from the traditional ones and they expose the public keys that participate in the transaction authorization, revealing the actors involved. Finally, the underlying blockchain has to support this solution, and therefore, this approach is not suitable for every crypto-asset.

Solutions similar to multisignature have also been devised for blockchains that do not natively support it. For example, in Ethereum [23] there are some smart contracts that replicate the same functionality. Unsurprisingly, this solution shares the same concerns of the cost and privacy of the Bitcoin multisignature: using the smart contract requires payment in gas and the data used by a smart contract is public, so the individual addresses that control the asset are exposed. Moreover, smart contracts (as all software) are likely to contain bugs, but in contrast to traditional software, they are immutable, thus there is little to no room for fixing possible problems. A relevant example is given by the *Parity Bug* [24] in Ethereum. A library developed for multisignature wallets was initialized as a contract at the address `0x863df6bfa4469f3ead0be8f9f2aae51c91a907b4`. On 19 July 2017, a bug in a pair of functions for the multisignature protocol setup was exploited by an unknown attacker to steal 153 037 ETH [25]. In the new version, released after the attack, there was a new bug: the library had been initialized without any owner. Since Ethereum does not distinguish between accounts, libraries, and contracts, on 6 November 2017 it was possible for the GitHub user `devops199` to send a first transaction to become the owner of the library and a second transaction to kill the library. As a result, nearly 600 different wallet owners who were using that library have not been able to withdraw their funds, and 513 736 ETH have been frozen forever.

For further readings about risks and vulnerabilities concerning smart contracts, see [26,27].

3. Threshold Multiparty Computation (MPC) Signatures

The idea described in Section 2.5 of sharing the responsibility of custody has been further developed in recent years with threshold-signature solutions based on *multiparty computation* (MPC). The private key of an account is substituted by multiple partial keys, which are equivalent to shares of the centralized private key. These partial keys are held by various actors who generated them with a collaborative interactive protocol. When a number of actors at least equal to a fixed threshold agree, they can perform a signature. In particular, the signature is produced following a (usually interactive) protocol of multiparty computation, where the account's private key is never created nor reconstructed.

At first glance, this kind of solution sounds very similar to standard multisignature schemes, but they are actually quite different:

- *Final single signature*. In standard multisignature schemes, in order to authorize a multisignature transaction it is necessary that several actors use their individual private keys to affix their respective signatures. In the MPC threshold schemes, instead, the involved actors simultaneously use their key shares to interactively compute a final single signature that is equivalent to a "standard" digital signature. The compatibility of the signature with the standard algorithm means that this custody solution does not depend on blockchain support, namely, it is blockchain-agnostic. As a result, it can be applied to many current prominent or in-progress (e.g., Libra) cryptocurrencies that do not natively support multisignature schemes. In other words, this threshold signature scheme is able to introduce the overall multisignature concept even in cryptocurrencies that rely only on single signatures. Moreover, it is not possible to establish which shares of the key have been used to sign. In other words, unlike multisignature, we do not know who actually signed a transaction. This leads, on one hand, to increased privacy, and on the other hand, to a lack of accountability, which in some contexts may be very desirable. However, this approach has a major drawback: the actors involved in the signature must be online at the same time and communicate with each other to perform the computations, whereas with multisignature the individual signatures can be computed independently and put together later.
- *Indistinguishable transactions*. While with multisignature-like systems "special" public addresses are required, with MPC the public key involved in the resulting signature is just like any other standard, single-signature, public key. Thus, an observer cannot distinguish a transaction made by multiple parties and a transaction made by a single party. So the privacy of the signing parties is greatly improved.
- *Off-chain approach*. These kinds of MPC schemes work entirely off-chain, and an immediate consequence is having, in general, lower costs than on-chain solutions. Furthermore, being off-chain, there is the possibility of relying on custodians outside the blockchain world for custody and backup solutions.

Also note that this solution is *trustless*, since no final private key is ever created, nor can one be created by any single party. Therefore, there is no need to trust anyone with the private key generation at any point in time. This is a fundamental difference with respect to using a simple secret-sharing scheme to split the private key. In fact, using a secret-sharing scheme would create two single-points of failure: i.e., moments in which at least one party would know the entirety of the private key (and therefore gain complete control). The first critical moment of a secret-sharing scheme is during the key generation phase, since a single party generates the key and then computes and distributes the shares; the second critical moment of a secret sharing scheme is during the signing phase, when at least one entity must collect the various shares and reassemble the key to perform the signature.

Given the aforementioned advantages, MPC signatures have attracted a lot of research interest and effort. A critical point has been the development of an efficient key generation in the context of more than two parties, until the breakthroughs of Gennaro and Goldfeder [28] and Lindell, Nof, and Ranellucci [29].

3.1. A (2,3)-Threshold MPC Signature Scheme

In [28], Gennaro and Goldfeder presented a (t, n)-*threshold* MPC signature protocol that allows distributing the signing process among a group of t parties, producing a t-of-n digital signature for ECDSA that can be verified by anyone with the public key.

In this section, we will outline the scheme for a very specific case, where there are three parties and the signing threshold has been set to two. In other words, we are considering a 2-of-3 version of the general scheme. For the sake of notation, we will denote the three actors by A, B, and C.

The scheme starts with each party generating a secret u_i and sharing it with a $(2, 3)$-threshold secret sharing scheme (a simplified version of Feldman's verifiable protocol [30]). In this way, each party has three shards $\sigma_{i,j}$ of their secret: one is kept secret, while the other two are sent to the other two parties, as summarized in Figure 1.

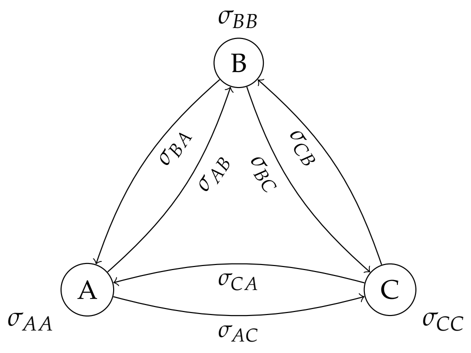

Figure 1. Sharing distribution with the $(2, 3)$-threshold secret-sharing scheme.

The value (never actually computed) that would correspond to the private key of the digital signature algorithm is:

$$x := u_A + u_B + u_C. \tag{1}$$

Then each party computes the value x_i as the sum of the two shards received and the one they previously kept secret. We can note that each x_i is a $(2, 3)$-share of the unknown value x.

When two out of the three parties want to sign a message with digest m, they follow these steps:

1. They convert the $(2, 3)$-threshold shares x_i of x into $(2, 2)$-threshold shares w_i of x with appropriate Lagrangian coefficients.
2. They randomly pick two values k_i, γ_i, that will be considered as $(2, 2)$-threshold shares of two values, say k and γ, which they both do not know.
3. Now the most delicate step: each party uses a *multiplicative-to-additive* share conversion protocol (MtA) exploiting the additive homomorphic properties of Paillier's cryptosystem [31], in order to get two $(2, 2)$-threshold shares (say δ_i and σ_i) of the values

$$\delta := k\gamma \quad \text{and} \quad \sigma := kx, \tag{2}$$

   for which they respectively own the shares k_i, w_i, and γ_i. Then, the value δ is reconstructed by both parties as $\delta = \delta_1 + \delta_2$.
4. Both parties compute the inverse δ^{-1} of the value δ, and they obtain $(2, 2)$-threshold shares of the value k^{-1} as $\delta^{-1}\gamma_i$.
5. They compute the value $\mathcal{R} = \gamma_1\delta^{-1}\mathcal{B} + \gamma_2\delta^{-1}\mathcal{B} = k^{-1}\mathcal{B}$, where $\mathcal{B}$ is the base point of the elliptic curve group used.
6. Finally, they compute $r := \mathcal{R}_x$ (i.e., the first coordinate of the point $\mathcal{R}$) and $s_i := mk_i + r\sigma_i$. Then they share s_i and compute the value:

$$s := s_1 + s_2. \tag{3}$$

The signature is the couple (r, s) and is completely compatible with the standard ECDSA digital signature.

$$s = k(m + rx), \tag{4}$$

and it can be verified using the public key $\mathcal{Y} = x\mathcal{B}$ (obtained as $\mathcal{Y} = u_1\mathcal{B} + u_2\mathcal{B} + u_3\mathcal{B}$, where each $u_i\mathcal{B}$ has been shared in the key generation phase) in the usual way: the signature is valid if and only if

$$(ms^{-1}\mathcal{B} + rs^{-1}\mathcal{Y})_x = r. \tag{5}$$

In the description of the above protocol, we omitted many details, for the sake of readability. So now we are going to discuss some crucial aspects.

3.1.1. Commitments

Let us start with commitments. They are placed nearly everywhere in the protocol, and they are used to let a party commit to a value while keeping it secret and revealing it only later on.

A commitment scheme is mainly used during a communication in order to ensure that the parties cannot cheat and base their choices on the values sent by the other parties. In fact, in a practical scenario we cannot assume that the communication is simultaneous.

A possible implementation of a commitment scheme uses a secure hash function as the commitment function, which processes the value to be committed m concatenated with a random value r. In this case, the decommitment string is this concatenation of m and r and the decommitment function checks that its digest is equal to the commitment string. If the test is passed, then the function outputs the value m (truncating the decommitment string), otherwise it fails. A failure signals that something went wrong or the other party tried to cheat, so it causes the protocol to abort to preserve the security.

3.1.2. Zero-Knowledge Proofs

A Zero-Knowledge (ZK) proof is an interactive protocol between a prover and a verifier, where the former wants to convince the latter that they know some secret information (proof of knowledge) or that a proposition of the form $p \in P$ is true (proof of membership), without revealing anything about their secret knowledge to the verifier.

In the described MPC multisignature protocol, many ZK proofs are used:

- *Proof of knowledge of integer factorization* (taken from [32]) has been used in the context of the Paillier cryptosystem. In particular, it is used by each party to prove to the other that they know the factorization of the modulus n associated with the public key.
- A *"standard" range proof* is used during the multiplicative-to-additive protocols, alongside another ZK proof specially designed for this purpose. In particular, in order to ensure the correctness of the MtA protocol, it is necessary that both participants prove that their shares are smaller than a certain threshold K.
- *Schnorr's proof* (taken from [33]) and one of its variants are used throughout the protocol when a party is broadcasting some g^x, to prove that they know x.

3.2. MPC Signatures with an Offline Recovery Party

Longo, Meneghetti, and Sala in [34] (and in [35,36] with more complete and formal presentations that include the security proofs) have used the protocol just described in Section 3.1 as a starting point to develop a $(2, 3)$-threshold multisignature protocol with a useful property: one of the three parties, which we will call the *Recovery Server*, does not need to participate during the key generation step, which normally would involve all three parties in a (2,3)-threshold scheme. This actor gets involved only when one of the remaining parties is not able to sign. In that case, they receive all the information needed from the remaining online party, and then they both proceed to sign.

This feature is the main improvement over [28], and it gives considerable benefits in real-life applications of the protocol because if the recovery party can stay offline the vast majority of the time (and crucially during the enrollment of new users). then it can implement hardened security features (e.g., air-gapped cold storage) whilst preserving a scalable and efficient infrastructure.

In more formal terms, the protocol involves three actors:

- *The Service Provider*, who is always online.
- *The Recovery Server*, who has to be online only at the beginning, and afterwards only if one of the other two parties is not able to sign.
- *The Client*, who is unknown at the beginning (i.e., unknown to the Recovery Server) and later enrolls and signs by interacting with the Service Provider.

The scheme is structured in four phases:

1. *Initialization phase*: the setup phase for the Service Provider and the Recovery Server, performed only once.
2. *Enrollment phase*: performed when a new Client wants to enroll in the system (the Recovery Server is not needed during this phase).
3. *Ordinary Signature phase*: performed when a Client wants to sign a message (e.g., a transaction). It requires the collaboration of the Service Provider, but again, the Recovery Server is not needed during this phase.
4. *Recovery Signature phase*: when either a Client or the Service Provider is unable to sign (e.g., due to the loss of the private key), the Recovery Server is brought back online to collaborate with the uncompromised party in order to perform the signature (e.g., transferring the funds to a new wallet).

Let us give an example of an application in the context of cryptocurrencies. Suppose that Carol, who will become the Client in our protocol, would like to open a wallet to make transactions with some cryptocurrency. She is also aware that, by handling a single private key on her own, she is exposed to risks (e.g., private key loss, no inheritance). Therefore, she chooses an MPC multisignature solution, which allows her to recover the funds even in case of the loss of her private key. For this reason, she turns to a third-party service, namely the Service Provider, which enrolls Carol as the Client. When Carol wants to transfer funds from her wallet, she prepares the transaction and signs it with the collaboration of the Service Provider. If Carol loses her private key, she contacts the Recovery Server so that it starts the recovery signature phase with the Service Provider, signing a transaction that moves all of her funds to a new address. In contrast, if the Service Provider loses the private keys it is protecting (maybe due to an attack or a system failure, or if the company goes bankrupt), then the Recovery Server performs the same recovery signature phase with each client. As a result, this scheme ensures that Carol's funds are not lost.

Now, let us recall the trade-off logic of security versus spendability that we introduced at the very beginning of Section 2. This scheme reconciles the two aspects: the user usually signs transactions with the Service Provider, which is always online and guarantees spendability. Meanwhile the Recovery Server guarantees security and safety as long as it does not collude with the Service Provider.

A practical application of this solution could exploit a public notary or a bank as the trusted offline third party (i.e., the Recovery Server). This requires a single preliminary arrangement between the Service Provider and the chosen trusted party, and later on multiple users can take advantage of it to protect their assets with much lower costs. Note that, unlike what was proposed in Section 2.4, here the notary is not a single point of failure and trust is shared with the service party, since both must collude to allow foul play. Moreover, these kinds of choices for the recovery server also solve the problem of inheritance since as the heirs accept the inheritance, the notary or the bank are able to transfer the funds to the heirs collaborating with the Service Provider.

The protocol enjoys all the properties of a generic threshold MPC signature scheme (see Section 3). In particular, we want to highlight what is effectively a blockchain-agnostic system, where multiple parties are involved and funds can be safely recovered in case of emergency. This signature is available for both ECDSA [35] and EdDSA [36] signatures, and so it can be applied to any cryptocurrency using ECDSA signatures (e.g., Bitcoin) or EdDSA signatures (e.g., Tezos, Stellar). Indeed, a practical use-case of this protocol can be found in Ethereum [37] (for ECDSA) and Libra [38] (for EdDSA) wallets. In addition, the scheme is completely compatible with a deterministic derivation of public and private keys (see Section 2.4), which is useful in contexts that need the generation of many keys from a single key, as is required in the Bitcoin Improvement Proposal 32 [39] (BIP32). Finally, the scheme integrates a mnemonic encoding of secret data that adds another layer of resiliency (see Section 3.2.4).

3.2.1. Let the Recovery Server Stay Offline

In this section, we will explain how the Recovery Server is able to stay offline during the Enrollment phase but still be able to sign when needed (even if it skipped the shards generation shown in Figure 1).

During the initialization phase, the Recovery Server generates a long-term private/public key pair for a given asymmetric cryptosystem and sends the public key to the Service Provider. After that, the Recovery Server is allowed to go offline.

For any new Client who wants to enroll in the system, the Service Provider is involved. Together, they perform the same steps shown in the previous section, with the following changes:

- At the very beginning, the Service Provider sends the Recovery Server's public key to the Client.
- In order to "surrogate" the shards creation by the Recovery Server (would have taken the role of the actor B), the Service Provider (A) and the Client (C) randomly pick $\sigma_{B,A}$ and $\sigma_{B,C}$, respectively.
- Both parties encrypt the shards pair $\sigma_{B,i}$, $\sigma_{i,B}$ with the Recovery Server's public key, and they exchange them among themselves. In this way, both parties have the information needed for the Recovery Server to eventually join the protocol.

Let us now explain how the recovery signature works. Without loss of generality, we can assume that the party that loses its private key material is the Client (that is also the most realistic scenario). In this case:

1. The Service Provider contacts the Recovery Server, which comes back online and receives the encrypted shards.
2. The Recovery Server decrypts the received shards and computes x_B, which should be the sum of the shards $\sigma_{A,B}$, $\sigma_{C,B}$ and $\sigma_{B,B}$. But the last one does not exist. This is not a problem since the shards $\sigma_{i,j}$ are effectively a $(2,3)$-threshold secret sharing of some secret u_i, so the information given by $\sigma_{B,A}$ and $\sigma_{B,C}$ is enough to compute a unique $\sigma_{B,B}$ that is compatible with this scheme. To sum up, the Recovery Server "generates" its secret u_B starting from the shares received.
3. Finally, the Recovery Server and the Service Provider can turn their secrets x_A and x_B into $(2,2)$-threshold shares w_A and w_B of x and proceed with an ordinary signature.

3.2.2. EdDSA Version

Regarding the EdDSA version, the protocol is quite different in the signature phase but shares the same ideas to let one party to stay offline. Let us briefly explain the differences with ECDSA:

- The enrollment phase is repeated twice, namely, at the end each party will have two secrets u_i, u'_i, two shares x_i, x'_i, and two shared values $\mathcal{A}$, $\mathcal{K}$.
- The idea behind the signature of a message M is the following:
    1. Both parties transform their shares from $(2,3)$-threshold into $(2,2)$-threshold shares: $x_i \rightarrow w_i$ and $x'_i \rightarrow k_i$.
    2. Each party computes $\mathcal{R}_i := k_i H(\mathcal{K}, M)\mathcal{B}$, and they exchange them (H is the hash function used in EdDSA, and $\mathcal{B}$ is the base point of the elliptic curve group).

3. Both parties can compute the first component of the signature as:

$$\mathcal{R} := \mathcal{R}_1 + \mathcal{R}_2 = (k_1 + k_2)H(\mathcal{K}, M)\mathcal{B}. \tag{6}$$

4. Each party computes the shard $S_i := k_i H(\mathcal{K}, M) + w_i H(\mathcal{R}, \mathcal{A}, M)$, and they exchange them.
5. Both parties can compute the second component of the signature

$$S := S_1 + S_2 = (r'_1 + r'_2)H(\mathcal{K}, M) + (w_1 + w_2)H(\mathcal{R}, \mathcal{A}, M). \tag{7}$$

Note that the signature can be verified by the standard EdDSA algorithm, and both parties verify it before completing the protocol successfully.

3.2.3. Key Derivation

During the enrollment phase, the Service Provider and the Client also generate a common secret $\mathcal{D}$ that the Recovery Server can recover by decrypting the encrypted shards. The threshold shares obtained in the Enrollment Phase can be utilized directly to sign messages, or they can be used to derive deterministically other key pairs using this common secret. For example, in Bitcoin it is good practice to always use fresh addresses, which correspond to different keys (see, for example, BIP32 [39]).

For this purpose, it is sufficient that A and C agree on a (public) derivation index i. Then, using the common secret $\mathcal{D}$ computed during the Enrollment Phase, they can independently derive the shards $w_A^{(i)}$ and $w_C^{(i)}$ and use them in the Signature Phases in place of w_A and w_C, respectively. Note that the derived secret keys correspond to a new public key $\mathcal{Y}^{(i)}$. The derivation can also be compound, that is, more keys can be derived from a derived key.

3.2.4. Mnemonic

The protocol allows saving critical private data as a list of common words, known as *mnemonic encoding*, that can be memorized and later used to reconstruct the original data. This allows avoiding the Recovery Signature Phase (and thus the cost of involving the Recovery Server), as long as the mnemonic is available. In a practical scenario, the usage of mnemonic encoding is quite convenient and less expensive since the Recovery Signature Phase leads to another Enrollment Phase as well as an Ordinary Signature Phase to transfer the funds to the new wallet.

The protocol generates the mnemonic (using an approach similar to the one used in Bitcoin's BIP39 [40]) from the secret parameters of a user and includes a few checksum bits extracted from the shared secrets and the public key. When the restoration of this data is needed (e.g., when the Client changes or upgrades the device that stored their secret material), the other uncompromised party provides the shared values whose integrity is verified using the checksum bits.

3.2.5. Security

The protocols presented in Section 3.1 and in this section are provable as secure in the standard model. More precisely, the signatures are proven unforgeable following the standard notion of existential unforgeability against chosen message attacks (EU-CMA) as introduced in [41], adapted for a threshold multiparty scheme.

Definition 1 (Existential Unforgeability of a (2,3)-Threshold Signature Protocol)**.** *Consider a malicious Probabilistic Polynomial Time (*PPT*) adversary $\mathcal{A}$ who participates in a (2,3)-threshold signature scheme S for the creation of a public key $\mathcal{Y}$ and in the creation of signatures on adaptively chosen messages. Let M be the set of messages queried by $\mathcal{A}$. The protocol is said to be existentially unforgeable if there is no such PPT adversary $\mathcal{A}$ that can produce a signature on a message $m \notin M$, except with negligible probability.*

The main results are the following:

Theorem 1. *Assuming that*

- *the ECDSA signature scheme is unforgeable;*
- *the strong RSA assumption holds;*
- $(\mathrm{Com}, \mathrm{Ver})$ *is a non-malleable commitment scheme;*
- *the Decisional Diffie Hellman Assumption holds;*
- *and that the encryption algorithm used by the Recovery Server is IND-CPA;*

the threshold ECDSA protocol is unforgeable.

Theorem 2. *Assuming that*

- H *is a cryptographic hash function with some good PRNG properties;*
- *the EdDSA signature scheme with parameters* $(p, b, H, c, n, a, d, \mathcal{B}, q)$ *is unforgeable;*
- $\mathrm{Com}, \mathrm{Ver}$ *is a non-malleable commitment scheme;*
- *the Decisional Diffie Hellman Assumption holds;*
- *the encryption algorithm used by the Recovery Server is IND-CPA;*

the threshold protocol built with the hash function H *is unforgeable.*

The proofs are omitted here but can be found in [35,36].

The addition of an offline party does not deeply impact the signature phase of the protocol, so the security proof in [35] is largely derived from [28]; however, the analysis considers the changes in the enrollment phase and proves that the tricks used to keep the recovery party offline do not compromise the security, and takes into account the differences when the adversary is online or not.

The theorems above are focused on the unforgeability of the signature; however, with an offline party (and more so in the application context of crypto-assets management), there is another security aspect worth considering: the resiliency of recovery in the presence of a malicious adversary. Of course, if the offline party is malicious and unwilling to cooperate in fund recovery, then there is nothing we can do about it; however, the security can be strengthened considering that one of the online parties may corrupt the recovery material. In this case, a generic CPA asymmetric encryption scheme is not sufficient to prevent malicious behavior because the protocol needs a verifiable scheme that allows the parties to prove that the recovery material is consistent, just like they prove that they computed the shards correctly.

3.3. Access Structures for Digital Signatures

An alternative approach to the scheme of Gennaro and Goldfeder [28] was proposed around the same time by Lindell et al. in [29], where the authors propose a threshold ECDSA-based signature scheme that replaces Paillier homomorphic encryption with ElGamal in the exponent, reaching similar results with different security assumptions.

Their signature system can be applied to solve the problem of custody of crypto-assets exactly as for the methods previously described.

An important feature that the authors highlight about their protocol is the fact that it easily supports very complex access structures for signatures, combining additive sharing and key replication. The case we have seen so far of the access structure is the following: we have a set U of n users, and we set a threshold $t \leq n$; they can sign only if at least t users of U agree. The access structure considered here is instead more generic: we have l sets of users $U_1, \ldots, U_l$ of size $n_1, \ldots, n_l$, respectively, and we fix the respective thresholds $t_1, \ldots, t_l$, one for each set of users. At this point, we write the necessary condition for signing as a boolean formula comprising AND and OR operators on these sets.

For example, by imposing $(A_1 \vee A_2) \wedge A_3$ as condition, we mean that at least one of the sets A_1 and A_2 must reach the threshold, and so must A_3.

Such complex access structures allow managing the trade-off between security and transaction time in a very fine way for each set of actors. For example, let a group be comprised of employees of a bank, then we can set for it a certain threshold, while we can set a higher threshold for a group of servers (since involving a group of human beings is usually more complicated).

3.4. Alternative Application of MPC Signatures: CHECKPOINT

Having presented the structure of MPC signatures and some protocols, we now discuss an interesting proposal introduced in [42], which makes an alternative use of threshold MPC signatures. Instead of simply applying the MPC scheme to enhance the signature, CHECKPOINT exploits an auxiliary blockchain and shifts all the computations on a group of custodians.

The actors involved are the user C and a fixed set of custodians $A_1, \ldots, A_n$. Upon enrollment of the user C, an MPC key generation is performed between the custodians, which leads to the creation of the shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ of a private key, associated to the user C, for a given blockchain (e.g., Bitcoin). The user C is instead provided with a different private key, sk, which is used to sign on an auxiliary blockchain. Let Carol be a user who wants to sign a transaction on the real blockchain, then she must use her private key sk to sign a transaction on the auxiliary one, in which she specifies the data of the transaction she wants to carry out (amount, recipient(s), . . .). As soon as the custodians see the transaction on the auxiliary blockchain, they perform an MPC threshold signature on the real blockchain, on behalf of Carol.

The additional intermediate signature brings a fundamental advantage over the other threshold MPC schemes: the user's device does not have to participate directly in the multiparty computation, greatly reducing the computations to be performed on the user side.

Furthermore, nothing prevents Carol from being able to manage more than one blockchain at a time, always keeping only one private key. It would be sufficient that, when signing on the auxiliary blockchain, Carol specifies on which blockchain she wants to sign the transaction, alongside the other data.

On the other hand, it is obvious that performing an intermediate signature, especially on another blockchain, can considerably slow down the total time for the real transaction to be completed.

A more critical point is that the burden of custodying the original wallet's key is simply transferred to the custody of the new key used to authorize transactions on the auxiliary blockchain, unless the system is integrated with other authentication mechanisms that allow a safe recovery.

4. Conclusions

In this paper, we analyzed the custody of crypto-assets as the problem of protecting the private key that controls said assets, both from possible attackers and from loss due accidents of various nature (system or device failure, theft or misplacement, human error or memory deficits, death of the key holder), without compromising the accessibility and transactional ease of use of the funds.

With this in mind, we presented several possible solutions that are largely in use, discerning between self-custody and third-party custody solutions, describing hot storage and cold storage, showing specific solutions for the problem of private key recovery and also the new approaches of threshold multisignature, remarking how each solution has to undergo a trade-off between security and usability, and we summarize the comparison of the schemes in Table 1.

So far, most financial institutions (e.g., Fidelity) have opted for a cold-storage approach: easier to implement and safeguard but impractical for transactions. As a result, such "static" solutions are not suitable for the incoming wave of stablecoins and digital currencies, which are to be used for transactional purposes [43].

We believe that the crypto-assets custody solutions of the future must be "dynamic". That is, they must provide the same level of security as the static ones, and at the same time allow for easy transactional use. Our belief is also echoed by the CEO of the Nomura-backed Komainu, whose custodial solution claims to "offer the same level of security as cold storage solutions while allowing for the speed and user flexibility of online hot wallets" [44].

Table 1. Qualitative comparison.

Method	**Pros**	**Cons**
Self-custody (in general)	• The user has complete control.	• The user bears full responsibility for security and safety.
Full third-party custody (e.g., exchange platforms)	• The user is not responsible for the security. • Usually high safety and ease of use.	• The service provider has to be fully trusted for honesty, security, safety practices, and existence of funds.
Hardware wallets	• Protection against online attacks. • Enhanced security compared to traditional storage. • The user retains direct control.	• A little bit slower and less easy to use compared to traditional storage. • The device may be lost or suffer faults. • The user may forget the PIN. • The user may die before telling the PIN or the device position to anyone else. • The private key may be stolen at the moment of the signature.
Offline software wallet	• Protection against online attacks. • The user retains direct control.	• Low ease of use. • Similar problems to hardware wallets, possibly more subject to failures due to the greater complexity.
Notary as recovery service	• High safety. • Perfectly compatible with inheritance procedures.	• The user has to trust the notary. • Expensive.
Secret-sharing of private key	• Enhanced safety. • May be combined with other solutions to further enhance safety and security. • Great flexibility.	• Key vulnerable when signing. • Lowered ease of use since multiple parties or devices have to be involved.
On-chain multisignatures	• Enhanced security with no single point of failure. • Enhanced safety. • Accountability of transaction signers. • Ease of use helped by the non-interactive nature of the scheme.	• Higher cost of transaction fees. • Diminished privacy. • Blockchain-dependent solution.
Threshold MPC signatures	• High security with no single point of failure. • Enhanced safety. • More flexibility and lower fees due to off-chain approach. • Blockchain-agnostic. • Privacy: transactions and wallets are indistinguishable from standard ones.	• Complex protocols, especially key generation. • Ease of use hampered by interactive signature protocol. • No accountability.

Therefore, here we have described in more detail a couple of threshold signature schemes based on multiparty computation that show many desirable properties in their application to both custody and transactional use of cryptocurrencies. As a matter of fact, they are trustless, since an "almighty" private key is never created, and they overcome the limitations of blockchains that do not have native multisignature support. In addition, they allow reducing the transaction size and cost in blockchains that already support multisignature schemes (e.g., Bitcoin) and improve the overall level of privacy since the multiple signers are not exposed to public view and the address is indistinguishable from normal ones.

In particular, in Section 3.2 we presented a solution that consists in a 2-of-3 threshold signature scheme, which inherits all the properties of threshold signature schemes just listed and, in addition, allows one party to not participate in the key generation phase. This enhances the practicality of the protocol since one party is only rarely involved (just in case it is necessary to recover the crypto-asset). This means that the third actor can be a very secure and trustworthy entity that would be too cumbersome and/or expensive to involve in the enrollment of each and every user.

Future research on this topic may involve the development of a threshold MPC version of other digital signature schemes used to control crypto-assets. Some (e.g., Schnorr signatures that have been recently introduced in Bitcoin [33]) may be relatively easily derived from other protocols. Others, however, will require some effort (e.g., Ring Signatures used in Monero [45]). Further research in the field of MPC schemes may aim to speed up the whole protocol, especially improving the ZK proofs that are the heaviest component. One direction could be to investigate how to modify the scheme in order to reduce the number of ZK proofs; another one could be to look for improvements in the sub-protocol itself in terms of both computations and communication steps. Further research efforts may be spent aiming to reduce the overall number of communication rounds in the protocol, towards one-round protocols that could be run asynchronously.

Author Contributions: Conceptualization, V.D.N., R.L., F.M., and G.R.; Formal analysis, V.D.N., R.L., F.M., and G.R; Investigation, V.D.N., R.L., F.M., and G.R.; Methodology, V.D.N., R.L., F.M., and G.R.; Supervision, V.D.N., R.L., F.M., and G.R.; Validation, V.D.N., R.L., F.M., and G.R.; Writing—original draft, V.D.N., R.L., F.M., and G.R; Writing—review and editing, V.D.N., R.L., F.M., and G.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: For their suggestions and comments, we would like to thank Alessio Meneghetti, Massimiliano Sala, and Zsolt Levente Kucsván.

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

References

1. Frauenfelder, M. 'I Forgot My PIN': An Epic Tale of Losing $30,000 in Bitcoin. 2018. Available online: https://www.wired.com/story/i-forgot-my-pin-an-epic-tale-of-losing-dollar30000-in-bitcoin/ (accessed on 31 August 2020).
2. Fidelity Rolls Out Cryptocurrency Custody Business. Available online: https://www.ft.com/content/ca95d640-f0b6-11e9-ad1e-4367d8281195 (accessed on 28 September 2020).
3. Nomura and Partners Launch Digital Asset Custodian Komainu. Available online: https://www.reuters.com/article/us-crypto-currencies-nomura-idUSKBN23O2AS (accessed on 28 September 2020).
4. Italian Bank Releases Bitcoin Wallet Despite Coronavirus. Available online: https://decrypt.co/23034/italian-bank-releases-bitcoin-wallet-despite-coronavirus (accessed on 28 September 2020).
5. Coinbase—Buy & Sell Bitcoin, Ethereum, and More with Trust. Available online: https://www.coinbase.com/ (accessed on 31 August 2020).
6. Bitcoin & Cryptocurrency Exchange—Kraken. Available online: https://www.kraken.com/ (accessed on 31 August 2020).
7. Bitstamp—Buy and Sell Bitcoin. Available online: https://www.bitstamp.net/ (accessed on 31 August 2020).
8. Goud, N. Cyber Attack Disrupts Bitcoin Exchange Bitfinex Services! Available online: https://www.cybersecurity-insiders.com/cyber-attack-disrupts-bitcoin-exchange-bitfinex-services/ (accessed on 31 August 2020).
9. McMillan, R. The Inside Story of Mt. Gox, Bitcoin's $460 Million Disaster. 2014. Available online: https://www.wired.com/2014/03/bitcoin-exchange/ (accessed on 31 August 2020).
10. Hardware Wallet—State-Of-The-Art Security for Crypto Assets—Ledger. Available online: https://www.ledger.com/ (accessed on 31 August 2020).
11. Trezor Hardware Wallet—The Original & Most Secure Bitcoin Wallet. Available online: https://trezor.io/ (accessed on 31 August 2020).
12. KeepKey—Hardware Wallet—ShapeShift. Available online: https://shapeshift.com/keepkey (accessed on 31 August 2020).

13. Shane, D. A Crypto Exchange May Have Lost $145 million after Its CEO Suddenly Died. 2019. Available online: https://edition.cnn.com/2019/02/05/tech/quadriga-gerald-cotten-cryptocurrency/index.html (accessed on 31 August 2020).
14. de Candia, A. Charles Hoskinson: A Paper Wallet with $1 million in ADA. 2020. Available online: https://en.cryptonomist.ch/2020/08/07/charles-hoskinson-paper-wallet-million-in-ada/ (accessed on 31 August 2020).
15. Yubico—YubiKey Strong Two Factor Authentication. Available online: https://www.yubico.com/ (accessed on 31 August 2020).
16. Best Bitcoin Wallet Armory—Multi-Signature Cold Storage. Available online: https://www.bitcoinarmory.com/ (accessed on 31 August 2020).
17. Electrum Bitcoin Wallet. Available online: https://electrum.org/ (accessed on 31 August 2020).
18. Aydar, M.; Cetin, S.; Ayvaz, S.; Aygun, B. Private key encryption and recovery in blockchain. *arXiv* **2019**, arXiv:1907.04156.
19. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
20. Hamid, L. Biometric technology: Not a password replacement, but a complement. *BIOM Technol. Today* **2015**, *2015*, 7–10. [CrossRef]
21. Nakamoto, S. Bitcoin: A Peer-To-Peer Electronic Cash System. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 31 August 2020).
22. Andresen, G. M-Of-N Standard Transactions. 2011. Available online: https://en.bitcoin.it/wiki/BIP_0011 (accessed on 31 August 2020).
23. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
24. Technologies, P. A Postmortem on the Parity Multi-Sig Library Self-Destruct. 2017. Available online: https://www.parity.io/a-postmortem-on-the-parity-multi-sig-library-self-destruct/ (accessed on 31 August 2020).
25. Brenner, M. How I Snatched 153,037 ETH After A Bad Tinder Date. 2017. Available online: https://medium.com/@rtaylor30/how-i-snatched-your-153-037-eth-after-a-bad-tinder-date-d1d84422a50b (accessed on 31 August 2020).
26. Major Issues Resulting in Lost or Stuck Funds. 2018. Available online: https://github.com/ethereum/wiki/wiki/Major-issues-resulting-in-lost-or-stuck-funds#parity-multisig-library-contract-issue-2-4-5-6-7 (accessed on 31 August 2020).
27. Nikolić, I.; Kolluri, A.; Sergey, I.; Saxena, P.; Hobor, A. Finding the greedy, prodigal, and suicidal contracts at scale. In Proceedings of the 34th Annual Computer Security Applications Conference, San Juan, PR, USA, 3–7 December 2018; pp. 653–663.
28. Gennaro, R.; Goldfeder, S. Fast multiparty threshold ECDSA with fast trustless setup. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1179–1194.
29. Lindell, Y.; Nof, A. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1837–1854.
30. Feldman, P. A practical scheme for non-interactive verifiable secret sharing. In Proceedings of the 28th Annual Symposium on Foundations of Computer Science (sfcs 1987), Los Angeles, CA, USA, 12–14 October 1987; pp. 427–438.
31. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 223–238.
32. Poupard, G.; Stern, J. Short proofs of knowledge for factoring. In *International Workshop on Public Key Cryptography*; Springer: Berlin/Heidelberg, Germany,2000; pp. 147–166.
33. Schnorr, C.P. Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology*; Springer: Berlin/Heidelberg, Germany, 1989; pp. 239–252.
34. Longo, R.; Meneghetti, A.; Sala, M. Threshold Multi-Signature with an Offline Recovery Party. 2020. Available online: https://eprint.iacr.org/2020/023 (accessed on 28 September 2020).
35. Battagliola, M.; Longo, R.; Meneghetti, A.; Sala, M. Threshold ECDSA with an Offline Recovery Party. *arXiv* **2020**, arXiv:2007.04036.

36. Battagliola, M.; Longo, R.; Meneghetti, A.; Sala, M. A Provably-Unforgeable Threshold EdDSA with an Offline Recovery Party. *arXiv* **2020**, arXiv:2009.01631.
37. Di Nicola, V. Custody at Conio-Part 3. 2020. Available online: https://medium.com/conio/custody-at-conio-part-3-623292bc9222 (accessed on 31 August 2020).
38. Libra. Libra Developer Update: Core Roadmap #3. 2020. Available online: https://mailchi.mp/263f908c91f2/libra-dev-update-first-libra-core-summit-3759697 (accessed on 31 August 2020).
39. Wuille, P. Hierarchical Deterministic Wallets. 2012. Available online: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki (accessed on 31 August 2020).
40. Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. Mnemonic Code for Generating Deterministic Keys. 2013. Available online: https://github.com/bitcoin/bips/blob/master/bip-0039 (accessed on 31 August 2020).
41. Goldwasser, S.; Micali, S.; Rivest, R.L. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.* **1988**, *17*, 281–308. [CrossRef]
42. Private Key Recovery—Decentralize User's Responsibility. 2019. Available online: https://medium.com/iov-internet-of-values/private-key-recovery-decentralize-users-responsibility-88fa60ee905 (accessed on 31 August 2020).
43. Feds' "Stablecoin" Letter May Boost Crypto Ambitions of Facebook, Square. Available online: https://fortune.com/2020/09/22/occ-stablecoin-letter-crypto-currencuy-facebook-square/ (accessed on 28 September 2020).
44. Nomura-Backed Crypto Custody Venture Launches After 2 Years in the Works. Available online: https://finance.yahoo.com/news/nomura-backed-crypto-custody-venture-080324576.html (accessed on 28 September 2020).
45. Liu, J.K.; Wei, V.K.; Wong, D.S. Linkable spontaneous anonymous group signature for ad hoc groups. In *Australasian Conference on Information Security and Privacy*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 325–335.