Short Paper: Credential Loss Problem for Cryptowallet Custodians

Michele Battagliola[1] and Carlo Brunetta[2]

[1] Department Information Engineering, Marche Polytechnic University, Italy, m.battagliola@staff.univpm.it
[2] Independent Researcher, Bagnolet, France, brunocarletta@gmail.com

Abstract. The increasing adoption of cryptocurrencies poses the critical challenge of secure crypto-asset custody, where the loss of private keys results in irreversible fund inaccessibility. Unlike traditional banking, cryptocurrency wallets often lack recovery mechanisms, making credential loss a significant vulnerability.
This paper explores existing solutions adopted by custodians to mitigate this risk, including reliance on centralized mechanism, multi-signature wallets, and advanced multi signature schemes that compute threshold signatures. While centralization has high trust risks and multi signatures lack universal support, threshold signatures offer a promising distributed approach. The paper analyses the requirements for effective credential loss solutions, evaluates popular methods, and proposes tentative solutions aimed at reducing resource overhead. Furthermore, it outlines the design and algorithmic properties of a potential ideal solution.

Keywords: Threshold Signature · Cryptocurrency · Digital Signature

1 Introduction

The increasing diffusion of cryptocurrencies and blockchain technologies has driven significant advancements in cryptographic research, highlighting the critical importance of secure crypto-asset custody. Cryptocurrency wallets serve as digital repositories enabling users to manage their crypto-assets. These wallets usually rely on asymmetric cryptography: each user has a public key, which is usually referred as the *"address"* and it is used to receive transactions, and a private key, that is used to authorize transactions and grants ownership of the associated funds. The private key effectively represents absolute control over the digital assets held within the wallet and represents a critical point of failure: losing the private key could mean losing the entire content of the wallet.

Indeed, unlike traditional banking systems, where account recovery is often possible through centralized authorities, cryptocurrency wallets usually do not offer such solutions: once the private key is irretrievably lost, the associated digital assets become permanently inaccessible, effectively locking the funds away forever. This issue is exacerbated by the complexity of managing and securely storing these cryptographic credentials: users are highly vulnerable to human error, hardware failures, and attacks targeting private key theft.

The task of mitigating the risk of credential loss represents a critical challenge in the context of cryptowallet custody. Narayanan *et al.* [26, Sec. 4.2] highlights the security vulnerabilities of online wallets, emphasizing the need for offline solutions. A clear explanation of the challenges is provided by Dave [12].

In this context, the three main solutions investigated in literature are:

- to rely on a single trusted custodian that takes (full) responsibility of key management. This kind of centralization is usually against the *"spirit"* of blockchains, that have the decentralization as one of the main selling point. Moreover, this approach introduces a new single points of failure: these type of custodians are high value targets for criminal takeovers [9];
- to use multi-signature wallets (available for some cryptocurrencies, e.g. Bitcoin [25]) where the signatures are standard, but funds may be moved out of that wallet only when a sufficient number t of signatures corresponding to a prescribed set of n public keys (e.g. user and guardians) is provided. Unfortunately, this approach is not supported by every cryptocurrency (e.g. Ethereum [7]) and easily identifiable;
- to distribute the control of the wallet through advanced multi-signature schemes, in particular with threshold-like policies. Briefly, a (t, n)-threshold signature scheme (TSS) enables a distributed signing protocol among n players such that any subset of size t can sign, whereas any subset with fewer players cannot. The signature correspond to a *single* public key of which secret is shared among n parties, i.e. user and custodians. This solution is usually pursued by using a threshold signature compatible with the corresponding centralized one, to ensure compatibility and indistinguishability.

Our work focuses on the latter, i.e. securing the user's secret shard in event of a credential loss for custodian's solutions based on threshold signature schemes.

1.1 Related Works.

In the context of blockchain applications, the most important families of threshold signature are ECDSA, EdDSA and $\mathsf{Schnorr}$ signature. The first ECDSA threshold signature protocol was proposed by Gennaro *et al.* [17] where $t + 1$ parties out of $2t + 1$ are required to sign a message. Later, MacKenzie and Reiter proposed and then improved another scheme [24, 23], which has later been further enhanced [13, 14, 21]. The first scheme supporting a general (t, n)-threshold was proposed by Gennaro *et al.* [16], improved by Boneh *et al.* [6] and Gennaro *et al.* [15]. A parallel approach has been taken by Lindell *et al.* [22]. Kondi *et al.* [20] introduce a refresh mechanism, for proactive security against the corruption of different actors in time, that does not require all parties to be online, while Canetti *et al.* [8] take a similar approach and propose a protocol that streamlines signature generation and include proactive security mechanisms.

About $\mathsf{Schnorr}$, of particular relevance are Sparkle [11], EC:BLTWZ24 [1], and the FROST family [19, 10, 5]. Finally, several works [4, 2, 3] propose $(2, 3)$

versions of the ECDSA, EdDSA and $\mathsf{Schnorr\ TSS}$ that are specifically aimed to solve the problem of credential loss. In these protocols, one of the party (a designated recovery-custodian) can remain online for the whole key generation process and only return online when required in the credential recovery process. In this way, the burden on the recovery-custodian is drastically lowered.

1.2 Our Contributions.

The goal of this paper is twofold: firstly, we provide an overview of the critical requirement that a potential solution to the credential loss problem should satisfy. We analyse some of the most popular solutions highlighting their strength and weakness. Lastly, we provide tentative solutions to the problem, that aims to minimize the bandwidth, memory and communication rounds required but still lack to fully solve the problem. Therefore, we describe a potential ideal solution design and the algorithmic properties it must achieve.

2 Credential Loss Problem

The credential loss problem in the crypto-wallet custodian scenario is described as the problem of letting a user U to backup its secret shard z and provide such backup c to a different node S, which we call storer, for allowing future recovery in the event of U's loss of the secret. The storer S represent a valuable target for an attacker, so for safety reason the backup strategy should consider the eventuality of S being corrupted. For this reason, S is considered a malicious actor with the goal to obtain the secret shard from the backup meaning, formally, the security should hold against at least an honest-but-curious adversary or actively malicious, i.e. an adversary that actively deviate from the protocol, to any extend meaningful for the application's scenario.

We require a solution to such problem to achieve several requirements:

- ▷ *correctness*: if the backup c is correctly produced and the protocol between U and S is honest executed, the honest execution of the recovery protocol will always allow U retrieving its secret shard z;
- ▷ *identifiable misconduct*: both the backup and recovery protocols should allow a prompt identification of dishonest protocol's execution and the precise identification of the malicious actor misbehaving;
- ▷ *verifiable generation*: the backup generation protocol should let U provide formal reassurance π to S that the c is the correct computation of the backup of the secret shard z. The reason of such requirement is to guarantee to U honestly computing the backup that the recovery protocol will be successful. This implies that the correctness responsibility falls on S. Clearly, a maliciously computed backup would be immediately identified aborting the backup protocol thus guaranteeing S that only valid backup are stored.
- ▷ *verifiable storage*: the backup protocol should be considered terminate only when there is formal reassurance that the backup is correctly received and

stored by S and can be used during the recovery protocol to showcase S the correct reception of c. In other words, S must provide formal guarantees to U that the backup c was received which can be later used by U to highlight the successful backup generation and transmission.

These requirements interact to reassure both parties that the backup procedure is correctly computed and handled, even when the actors are untrusted and maliciously trying to maximize their profit at the other's expenses.

We suggest a formal notation for the abstract procedures required by a possible solution and depict in Figure 1 the general communication layout:

- $\mathsf{GenBackup}\left(\mathsf{z}, \mathsf{sk}_U, \mathsf{pk}_S\right) \to \mathsf{c}$: the generate backup procedure takes in input the secret to backup z, the users secret key sk_U, the storer public key pk_U and outputs the backup c which contains a generation proof π;
- $\mathsf{VerBackup}\left(\mathsf{c}, \mathsf{pk}_U, \mathsf{pk}_S\right) \to \{0,1\}$: the verify backup procedure takes in input the backup c containing a generation proof π, the public key of the user and the storer. The procedure outputs $\{0,1\}$ indicating the proof's correctness;
- $\mathsf{VerStorage}\left(\mathsf{c}, \mathsf{sk}_S, \mathsf{pk}_U\right) \to \sigma$: the storage verification procedure takes a backup c, the storer secret key sk_S and the user's public key pk_U, and outputs a proof of correct reception and storage of the backup to be verified by the user;
- $\mathsf{Recover}\left(\sigma, \mathsf{pk}_U, \mathsf{pk}_S\right) \to \mathsf{c}$: the recovery procedure takes the storage proof σ and the public keys of the two parties and outputs the backup c which allows the user to recover the secret z;

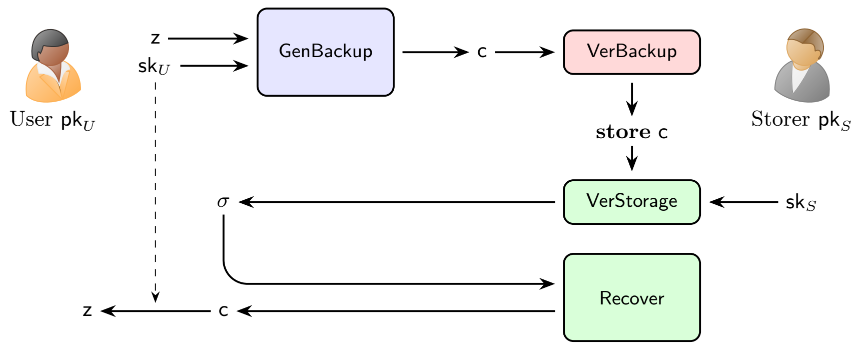

Fig. 1: General diagram of the backup and recovery procedures.

Any solution must accept the existence of natural problems introduced when considering the backup handling, e.g. the procedures required to securely handle the backup or the fact that even an honest storer might lose the backup by accident or because of hardware failure. Trivially, these events cannot be solved directly by the protocol but only by considering a more resilient backup strategy, e.g. using certified devices and cryptographic primitives and executing multiple redundant backup procedure with independent storer to increase the resilience.

Despite the importance in the applications, we leave as out-of-context the ability of incorporating a rewarding mechanism governing the backup procedure. Briefly, the backup and recovery procedure might be coordinated by a smart contract that enables an honest storer to be rewarded for their backup service or any actor can be punished whenever they do not follow the backup protocol correctly. These solutions are uncommon because of the protocol's complexity.

3 Tentative Solutions

We present two tentative solutions which partially solve the credential loss problem, since both come up short to be a fully satisfactory solution. Indeed, first we propose a very efficient and naive solution that lacks some critical verifiability guarantees. Afterwards, we propose a solution that offer high security guarantees but is rather inefficient. We denote with U the user, and with S the storer.

3.1 Naive Solution

The most naive solution is to merely use a symmetric encryption scheme (E, D) and let U provide as backup to S the encrypted secret share. Intuitively, U has a backup symmetric encryption key k that is safely stored long term. To create a backup, U encrypts its own share z with sk, i.e. $\mathsf{c}_U = \mathsf{E}(\mathsf{k}, \mathsf{z})$, and signs the resulting ciphertext $\mathsf{Sign}(\mathsf{sk}_U, \mathsf{c}_U)$ and obtains the signature σ_U. The backup is composed of the ciphertext and the signature $\mathsf{c} = (\mathsf{c}_U, \sigma_U)$.

The storer can verify the authenticity of the provided backup and reply with a receipt σ_S by signing the received backup. S must store the tuple $(\sigma_S, \mathsf{c}_U, \sigma_U)$ which is sent back to U when the recovery procedure is called. The recovery is trivially obtained by decrypting the backup $\mathsf{D}(\mathsf{k}, \mathsf{c}_U)$. Observe that the signatures might be generated with newly generated signing keys causing additional round of communication to exchange and authenticate such keys forcing a less practical protocol. Overall, the naive solution has clear advantages:

- ▷ the solution is easy to implement, fast and efficient. The backup is compact and the effective storage cost is dominated by twice the signature size which can be made quite small with an appropriate choice of a secure signing primitive;
- ▷ since the backup c is signed by S, U can be sure that S received the correct backup. When recovering, the same verification forces S to provide the same backup used during the receipt σ_S generation. These two verifications forces the S to store and retrieve the received backup;
- ▷ The security of U is perfect for an appropriate choice of the symmetric encryption scheme. For a secure primitive, S cannot learn anything from the ciphertext c_U and, as said above, he is sure that the ciphertext is the one created at the beginning;
- ▷ U cannot call multiple recovery procedure to gain additional shards. This is caused by the encrypted material being computed entirely by U, thus when the backup is retrieved back, the content does not lead to any additional knowledge.

Despite all the positive points, the solution completely lacks the ability to verify that the provided ciphertext c_U is indeed a correct encryption of the secret shard z, i.e. the suggested procedure $\mathsf{VerBackup}$. This is a problem since the user can encrypt the wrong data (maliciously or not) and even in the scenario of following the backup protocol correctly, the recovery of the secret shard would be impossible. This failing scenario identifies the wrongdoing responsibility on U. However, this is undesirable for real-world application since it implies a complete access loss to the user's wallet.

3.2 Knowledge Extractor Solution

The second proposed solution aims to solve the verifiability problem of the naive solution, ensuring that U encrypted the real secret shard z at a higher computational and storage cost.

The main idea is to exploit the algebraic relation between the user's public key pk_U and his secret share $\mathsf{z} = \mathsf{sk}_U$ to build an online-extractable zero knowledge proof in such a way that the proof's transcript can be used as recovery material. More specifically, U generates an asymmetric encryption scheme (e.g. ElGamal) backup key-pair $(\mathsf{sk}, \mathsf{pk})$. Let Π be a sigma protocol that allows U to prove the knowledge of z, with pk_U public and let Ch be the challenge space of Π. We require that Π is special sound which, in practice, is not a restrictive request since most known sigma protocol satisfies this property. The interactive backup procedure between U and S is defined as:

1. U computes the first message $\mathtt{cmt}$ of Π as intended;
2. for every possible challenge ch_i in Ch, U computes the corresponding response rsp_i, picks a random nonce r_i and computes $t_i = \mathsf{Enc}\,(\mathsf{pk}, \mathsf{rsp}_i; r_i)$;
3. U sends all the $\{t_i\}_{i\in|\mathsf{Ch}|}$ to S;
4. S picks a vector of random challenges $\gamma \leftarrow\!\!\$\; \{0,1\}^{|\mathsf{Ch}|}$ and sends it to U;
5. U replies by revealing $(\mathsf{rsp}_{\gamma_i}, r_{\gamma_i})$ for each $i \in \{1, \ldots, |\mathsf{Ch}|\}$;
6. to verify the response, S checks that $t_{\gamma_i} \overset{?}{=} \mathsf{Enc}\left(\mathsf{pk}, \mathsf{rsp}_{\gamma_i}; r_{\gamma_i}\right)$ for each index $i \in \{1, \ldots, |\mathsf{Ch}|\}$ and then run the verification algorithm of Π on input $(\mathtt{cmt}, \mathsf{ch}_\gamma, \mathsf{rsp}_\gamma)$. If both checks are correct, then S computes the receipt. Otherwise, the backup is improperly generated.

The above interactive protocol can be turned into a non-interactive one using the Fiat-Shamir Transform or, with some subtle modification due to our needs, the Unruh transform [27]. The soundness error of the above protocol is $\frac{1}{|\mathsf{Ch}|}$, thus it is possible that multiple parallel repetition are needed to achieve an application's appropriate level of soundness. The backup is simply the transcript of the zero-knowledge protocol between U and S which is stored and provided whenever requested. To retrieve the secret shard z, U decrypts all the t_i received and, due to the special soundness of Π, extract z.

This knowledge extractor solution provides some clear advantages:

- ▷ the solution allows the usage of any zero-knowledge protocol Π that proofs the knowledge of the secret shard with respect to a public value, e.g. the secret shard $\mathsf{z} = \mathsf{sk}_U$ for the public key $\mathsf{pk}_U = \mathsf{sk}_U \cdot \mathcal{B}$;
- ▷ the backup procedure can be made both interactively or non-interactively thus better adapting to different application scenarios;
- ▷ the security of U is guaranteed by the choice of an appropriate encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$. The security requirement would be that a S cannot learn anything from the ciphertexts t_i. In addition, the zero-knowledge property of Π provides additional security guarantees for the revealed challenges;
- ▷ as before, U cannot call multiple recovery procedure to gain additional shards. The encrypted material is computed entirely by U, so receiving it back does not lead to any additional knowledge;
- ▷ U cannot produce wrong backup material. By construction, the backup material is a proof of knowledge of the secret shard z which is extracted during recovery using the special soundness.
- ▷ depending on the choices made for the sigma-protocol and encryption scheme, proving the UC-security of the backup might be simplified because of the more natural simulatability of the solution's mechanism.

The main solution's disadvantage is the direct inefficiency caused by the soundness guarantees. The transcript is longer than the naive solution, even when Π is compact, since U needs to compute all the possible response for all the possible challenges and the usage of an asymmetric encryption scheme.

Regardless, the main problem is that the protocol must be repeated to achieve the appropriate soundness guarantees. For example when $|\mathsf{Ch}| = 2$, one might require a $2^{-\lambda}$ soundness level where $\lambda \sim 80-128$ indicates the number of protocol's executions that must be made. When compared with the naive solution, the computational costs are clearly higher while the storage space required is approximatively two order of magnitude higher.

4 Future Directions

As previously discussed, the tentative solutions encountered unwanted trade-off. One solution prioritized storage and computational efficiency, achieving compact backup but, unfortunately, without providing verifiability assurances on the backup correctness, key requirement for the cryptocurrency custodian's applications. The second solution provides strong security guarantees and full verifiability at a significant space-intensive and computationally inefficient cost, posing practical limitations for real-world deployment.

We are therefore left with a central challenge to achieve simultaneously a compact and confidential backup, possibly based on encrypting the secret shard, and an efficiently verifiable proof of backup's correctness, e.g. a proof of correct encryption, without compromising either security or practicality. In other words, our ideal solution would be the efficient naive solution with the verifiable guarantees provided by the knowledge extraction one.

4.1 Defining the Ideal Solution

The critical challenge lies in finding a secure symmetric encryption scheme that allows an efficient zero-knowledge proof that the message encrypted is indeed the secret shards, i.e. the proof π must guarantee the knowledge of a secret shard z that is both the plaintext of a correct encryption ciphertext $\mathsf{c} = \mathsf{E}(\mathsf{k}, \mathsf{z})$ and the logarithm of the user's public key $\mathsf{z} \cdot \mathcal{B} = \mathsf{pk}_U$.

To address this challenge, the proof can be obtained using Succinct Non-interactive ARguments of Knowledge (SNARKs) or Scalable Transparent ARguments of Knowledge (STARKs). These advanced cryptographic tools offer the advantage of producing short, easily verifiable proofs, providing strong assurances about the correctness of the encryption and the logarithm knowledge. However, caused by the extensive generality of such tools, this approach comes with increased costs, both in terms of the computational resources required to generate the proof and the storage overhead due to the size of the proof itself which is incomparable with the naive solution's efficiency. Despite the fast development of more efficient SNARKs/STARKs, we believe that the ideal solution should have an ad-hoc design that maximizes efficiency without requiring too complex proving frameworks.

For this reason, we envision the ideal solution to use more *classical* Non-interactive Zero-Knowledge (NIZK) protocols, e.g. Schnorr or Chaum-Pedersen for the discrete logarithm, in conjunction with a symmetric encryption scheme from which it is possible to extrapolate particular algebraic behaviours, e.g. the somewhat homomorphic property of ElGamal.

During our investigations, we identify a key obstacle which is the need to encode the secret shard into a compatible algebraic object, e.g. a scalar into an elliptic curve point in the case of elliptic curve ElGamal. This encoding must simultaneously preserve a useful algebraic structure for efficient zero-knowledge proofs and allow for straightforward decoding back to the original secret. A simplified example with the goal of highlighting the issue especially in the elliptic curves scenario, we would like to have both the curve's algebraic group operation plus an easy method to evaluate the discrete logarithm. Clearly, this duality is not achievable with the known algebraic encoding, i.e. the mapping from x to $x \cdot \mathcal{B}$, or the Koblitz [18] elliptic curve encoding.

4.2 Conclusions

Future research, focused on identifying an implementable ideal solution, should prioritize exploring novel encryption methods that allow for verifiable proof of correct encryption using classical zero-knowledge protocols, thereby circumventing the complexities and costs associated with SNARKs/STARKs. A crucial direction involves investigating innovative encoding techniques for secret shards into, for example, elliptic curve points. These techniques should maintain a rich algebraic structure suitable for protocols like Schnorr or Chaum-Pedersen, while also enabling efficient and direct decoding back to the original secret. This might involve exploring non-standard encoding maps or targeted modifications to existing encryption schemes.

Acknowledgements This research was partially funded by Palisade Inc.

References

1. Bacho, R., Loss, J., Tessaro, S., Wagner, B., Zhu, C.: Twinkle: Threshold Signatures from DDH with Full Adaptive Security. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part I. LNCS, pp. 429–459. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-58716-0_15
2. Battagliola, M., Galli, A., Longo, R., Meneghetti, A.: A Provably-Unforgeable Threshold Schnorr Signature With an Offline Recovery Party. In: Pizzonia, M., Vitaletti, A. (eds.) Proceedings of the 4th Workshop on Distributed Ledger Technology co-located with the Italian Conference on Cybersecurity 2022 (ITASEC 2022), Rome, Italy, June 20, 2022. CEUR Workshop Proceedings, pp. 60–76. CEUR-WS.org (2022). https://ceur-ws.org/Vol-3166/paper05.pdf
3. Battagliola, M., Longo, R., Meneghetti, A., Sala, M.: Provably Unforgeable Threshold EdDSA with an Offline Participant and Trustless Setup. Mediterranean Journal of Mathematics **20**(5) (2023). https://doi.org/10.1007/s00009-023-02452-9
4. Battagliola, M., Longo, R., Meneghetti, A., Sala, M.: Threshold ECDSA with an Offline Recovery Party. Mediterranean Journal of Mathematics **19**(1) (2021). https://doi.org/10.1007/s00009-021-01886-3
5. Bellare, M., Crites, E.C., Komlo, C., Maller, M., Tessaro, S., Zhu, C.: Better than Advertised Security for Non-interactive Threshold Signatures. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, pp. 517–550. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15985-5_18
6. Boneh, D., Gennaro, R., Goldfeder, S.: Using Level-1 Homomorphic Encryption to Improve Threshold DSA Signatures for Bitcoin Wallet Security. In: Lange, T., Dunkelman, O. (eds.) LATINCRYPT 2017. LNCS, pp. 352–377. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25283-0_19
7. Buterin, V.: Ethereum: a next generation smart contract and decentralized application platform, https://github.com/XXX:Buterin13/wiki/wiki/White-Paper (2013).
8. Canetti, R., Makriyannis, N., Peled, U.: UC Non-Interactive, Proactive, Threshold ECDSA, Cryptology ePrint Archive, Report 2020/492 (2020). https://eprint.iacr.org/2020/492.
9. Chohan, U.: The Problems of Cryptocurrency Thefts and Exchange Shutdowns. SSRN Electronic Journal (2018). https://doi.org/10.2139/ssrn.3131702. http://dx.doi.org/10.2139/ssrn.3131702
10. Crites, E., Komlo, C., Maller, M.: How to Prove Schnorr Assuming Schnorr: Security of Multi- and Threshold Signatures, Cryptology ePrint Archive, Report 2021/1375 (2021). https://eprint.iacr.org/2021/1375.
11. Crites, E.C., Komlo, C., Maller, M.: Fully Adaptive Schnorr Threshold Signatures. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, pp. 678–709. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-38557-5_22
12. Dave, M.: The Non-Custodial Myth: Why Your "Self-Custody" Wallet Might Not Be What You Think, (2025). https://themanthan.substack.com/p/the-non-custodial-myth-why-your-self (visited on 08/23/2025).

13. Doerner, J., Kondi, Y., Lee, E., shelat, a.: Secure Two-party Threshold ECDSA from ECDSA Assumptions. In: 2018 IEEE Symposium on Security and Privacy, pp. 980–997. IEEE Computer Society Press (2018). https://doi.org/10.1109/SP.2018.00036
14. Doerner, J., Kondi, Y., Lee, E., shelat, a.: Threshold ECDSA from ECDSA Assumptions: The Multiparty Case. In: 2019 IEEE Symposium on Security and Privacy, pp. 1051–1066. IEEE Computer Society Press (2019). https://doi.org/10.1109/SP.2019.00024
15. Gennaro, R., Goldfeder, S.: Fast Multiparty Threshold ECDSA with Fast Trustless Setup. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018, pp. 1179–1194. ACM Press (2018). https://doi.org/10.1145/3243734.3243859
16. Gennaro, R., Goldfeder, S., Narayanan, A.: Threshold-Optimal DSA/ECDSA Signatures and an Application to Bitcoin Wallet Security. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 16International Conference on Applied Cryptography and Network Security. LNCS, pp. 156–174. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39555-5_9
17. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust Threshold DSS Signatures. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, pp. 354–371. Springer, Berlin, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_31
18. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation **48**(177), 203–209 (1987). https://doi.org/10.1090/s0025-5718-1987-0866109-5
19. Komlo, C., Goldberg, I.: FROST: Flexible Round-Optimized Schnorr Threshold Signatures. In: Dunkelman, O., Jacobson, M.J., O'Flynn, C. (eds.) SAC 2020. LNCS, pp. 34–65. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-81652-0_2
20. Kondi, Y., Magri, B., Orlandi, C., Shlomovits, O.: Refresh When You Wake Up: Proactive Threshold Wallets with Offline Devices. In: 2021 IEEE Symposium on Security and Privacy, pp. 608–625. IEEE Computer Society Press (2021). https://doi.org/10.1109/SP40001.2021.00067
21. Lindell, Y.: Fast Secure Two-Party ECDSA Signing. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, pp. 613–644. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_21
22. Lindell, Y., Nof, A.: Fast Secure Multiparty ECDSA with Practical Distributed Key Generation and Applications to Cryptocurrency Custody. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018, pp. 1837–1854. ACM Press (2018). https://doi.org/10.1145/3243734.3243788
23. MacKenzie, P., Reiter, M.K.: Two-party generation of DSA signatures. International Journal of Information Security **2**(3–4), 218–239 (2004). https://doi.org/10.1007/s10207-004-0041-0
24. MacKenzie, P.D., Reiter, M.K.: Two-Party Generation of DSA Signatures. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, pp. 137–154. Springer, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_8
25. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Tech. rep., Manubot (2019)
26. Narayanan, A., Bonneau, J., Felten, E., Miller, A., Goldfeder, S.: Bitcoin and cryptocurrency technologies: A comprehensive introduction. Princeton University Press (2016)
27. Unruh, D.: Non-Interactive Zero-Knowledge Proofs in the Quantum Random Oracle Model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS,

pp. 755–784. Springer, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_25