Wamu: A Protocol for Building Threshold Signature Wallets Controlled by Multiple Decentralized Identities

David Semakula
hello@davidsemakula.com
https://davidsemakula.com

15th May, 2023

Contents

1. Introduction

Multisig wallets (e.g. Safe [1]) are already widely adopted [2] and have proven the importance of noncustodial shared wallets with threshold access structures controlled by multiple decentralized identities, for mainstream users and decentralized teams and organizations.

However, threshold signature wallets have some unique benefits over multisig wallets including: cost-effectiveness, universal interoperability, and enhanced privacy and security.

[1] Safe. https://safe.global

[2] Dune Analytics. [Mainnet] Safe. https://dune.com/safe/ethereum

This is because while multiple parties each independently sign a transaction and the set of signatures is evaluated against the access structure/security policy on-chain for multisig wallets, threshold signature wallets instead use a threshold signature scheme to allow multiple parties to jointly compute a single signature that's similar to those computed by traditional wallets (e.g. Metamask [3]).

1.1. Problem

Despite the aforementioned benefits, there is currently no mainstream threshold signature wallet alternative to multisig wallets for decentralized teams and organizations that require noncustodial shared wallets with threshold access structures because:

- Most mainstream threshold signature wallets (e.g. ZenGo [4] and Torus [5]) are designed for the single-user setting with each party simply being either a separate device or authentication factor for the same user.
- Most institutional threshold signature wallet solutions (e.g. Fireblocks [6], Sepior [7] and Taurus [8]) have architectures that are either infeasible and/or undesirable for decentralized teams and organizations because of one or more of the following requirements:
  - Centralized or trust-based identity infrastructure for authenticating signing parties.
  - Controlled network environments with low latency and/or persistent synchronous connections between signing parties.

1.2. Solution

The ecosystem needs a new breed of noncustodial threshold signature wallet solutions that are controlled by multiple decentralized identities and can run on mainstream consumer devices making them well suited for use by decentralized teams and organizations, and mainstream users.

Recent breakthroughs in threshold signing research have yielded non-interactive threshold signature schemes (e.g. CGGMP20 [1], GG20 [2] and CMP20 [3]) that allow for asynchronous communication between signing parties, making the use of mainstream consumer devices as signing parties viable.

To remove the need for centralized and/or trust-based identity systems, and provide a user experience similar to existing multisig wallets, Wamu introduces a unique approach of augmenting a state-of-the-art non-interactive threshold signature scheme (e.g. CGGMP20 [1]) by cryptographically associating each signing party with a decentralized identity. This is achieved by:

[3] MetaMask. https://metamask.io
[4] ZenGo. https://zengo.com
[5] Torus. https://tor.us
[6] Fireblocks. https://www.fireblocks.com
[7] Sepior. https://sepior.com
[8] Taurus. https://www.taurushq.com

- Splitting the secret share for each party between the party and the output of a signing operation by its associated decentralized identity thus making the signing operation a requirement for reconstructing the party's secret share as described in section 3.
- Adding peer-to-peer decentralized identity verification to the key generation and signing protocols (and optionally to the key refresh protocol) of the threshold signature scheme.
- Defining protocols for identity rotation, share addition and removal, threshold modification and share recovery (as described in section 4) that build on top of the above 2 augmentations.

NOTE: For interoperability with existing wallet solutions, the only requirement for decentralized identity providers is the ability to compute cryptographic signatures for any arbitrary message in such a way that the output signature can be verified in a non-interactive manner.

2. Preliminaries

The rest of this document describes how Wamu's unique share splitting and reconstruction, and share recovery protocols work. For these descriptions, we'll use the following notation:

- P denotes a party.
- I denotes a decentralized identity.
- sk denotes the secret key of a decentralized identity.
- Sig denotes a signing algorithm.
- q denotes the prime order of cyclic group of the elliptic curve.

NOTE: While the share splitting and reconstruction protocol is described in technical detail in this document, for simplicity, the share recovery protocol is only described at a high-level and no technical detail is provided for decentralized identity verification and rest of Wamu's sub-protocols. We refer the reader to Wamu's technical specification for the technical details that are not provided in this document.

3. Share Splitting and Reconstruction

Assuming that we have a secret share x for a party P with an associated decentralized identity I, the share splitting and reconstruction protocol describes how to split x between P and the output of a signing operation Sig by I so that the output of Sig is required to reconstruct the secret share x.

This is achieved by generating a message m (we'll refer to this message as the "signing share") and computing a "sub-share" β (i.e a share of the secret share x) in such a way that m needs to be signed by I using Sig to produce another "sub-share" α, such that α and β are shares of x under Shamir's secret-sharing scheme [4].

NOTE: Share splitting and reconstruction is a single-party localized concern that happens after (and is not related to) the distributed key generation (DKG) protocol of the threshold signature scheme.

3.1. Share splitting

Given a secret share x as input and access to the decentralized identity I with secret key sk, the share splitting protocol proceeds as follows:

1. Sample a random message m (i.e. the signing share).
2. Compute a signature $(r, s) = Sig(sk, m)$.
3. Compute the first sub-share of x as the point $\alpha = (r, s \bmod q)$.
4. Generate a line L (i.e a polynomial of degree 1) such that α is a point on the line and x is the constant term (i.e. Polynomial Interpolation [5])
5. Compute another point β from L such that $\beta \neq \alpha$, β becomes the second sub-share of x.
6. Erase both α and L from memory.
7. Return the signing share m and the sub-share β.

3.2. Share reconstruction

Given a signing share m and a sub-share β as input (i.e. the outputs of the share splitting protocol in section 3.1 above) and access to the decentralized identity I with secret key sk, the share reconstruction protocol proceeds as follows:

1. Compute a signature $(r, s) = Sig(sk, m)$.
2. Compute a sub-share α as the point $\alpha = (r, s \bmod q)$.
3. Generate a line L by performing Polynomial Interpolation [5] using α and β as inputs.
4. Compute x as the constant term of L.
5. Erase both α and L from memory.
6. Return x as the secret share.

NOTE: For ECDSA signatures, the value of the parameter s in $(r, s) = Sig(sk, m)$ is already computed modulo q. We use the notation $\alpha = (r, s \bmod q)$ for the sub-share to make it clear (at a glance) that the sub-shares are computed using finite field arithmetic.

4. Share Recovery

Share recovery is only possible if the user's decentralized identity either survived or can be recovered after the disastrous event. In either case, there are two options for share recovery depending on:

- A quorum of honest parties surviving the disastrous event.
- A backup (preferably encrypted) of a signing share m and sub-share β pair on user-controlled secondary or device-independent storage.

4.1. Share recovery with a surviving quorum of honest parties

If a quorum of honest parties survives the disastrous event, share recovery can be accomplished based on peer-to-peer decentralized identity verification.

The party P_i that needs to recover its secret share initiates a signature-authenticated share recovery request leveraging its associated decentralized identity I_i. The surviving quorum of honest parties collectively verify the request, and then initiate the key refresh protocol of the threshold signature scheme with P_i participating if I_i matches a previously verified decentralized identity for a signatory.

4.2. Share recovery with a backup on user-controlled secondary or device-independent storage

4.2.1. Overview of share recovery with backup From the share splitting and reconstruction protocol in section 3 above, we note that for any party P, the combination of a signing share m and a sub-share β alone is insufficient to reconstruct the secret share x. This is because a signature of m from the decentralized identity I is required to compute the sub-share α, so that α and β can then be used to reconstruct L and compute the secret share x as the constant term of L.

Therefore, a signing share m and sub-share β pair can be safely backed up to user-controlled secondary (e.g. a secondary device or a flash drive) or device-independent storage (e.g. Apple iCloud [9], Google Drive [10], Microsoft OneDrive [11], Dropbox [12] e.t.c) without exposing the secret share.

4.2.2. Share recovery with an encrypted backup For increased security, a signature of a standardized phrase can be used as entropy for generating an encryption secret which can then be used to encrypt the signing share m and the sub-share β using a symmetric encryption algorithm before saving them to back up storage. Share recovery would then start by signing this standardized phrase, using the signature to recreate the encryption secret and then decrypting the encrypted backup to retrieve the signing share m and the sub-share β.

4.2.3. Further security and usability considerations for share recovery For further improved security and usability, the signing share m can be prefixed with a custom message that alerts the user to the purpose of the signature. This can help reduce the effectiveness of an adversary that gains access to the backup and tries to trick the user into signing m.

[9] Apple iCloud. https://www.icloud.com.

[10] Google Drive. https://drive.google.com.

[11] Microsoft OneDrive. https://www.microsoft.com/en-us/microsoft-365/onedrive/online-cloud-storage.

[12] Dropbox. https://www.dropbox.com.

Additionally, it's possible to rerun the share splitting protocol to generate a new pair of a signing share m^* and a sub-share β^* such that $m^* \neq m$, $\beta^* \neq \beta$ and $L^* \neq L$ to be specifically used for backup and recovery. This gives us the option to have separate signing shares for backup and recovery with customized prefixes that make it clear to the user that they're signing a backup signing share.

Lastly, the backup signing share m^* can be generated based on user input (e.g. a passphrase or security questions) removing the need for it to be backed up together with a sub-share β^* but instead relying on the user to provide this input during recovery as a security-usability tradeoff.

5. Conclusion

The Wamu project (meaning "together") aims to unlock the benefits of threshold signature wallets for decentralized teams and organizations, and mainstream users that require noncustodial shared wallets with threshold access structures by:

- Defining an open protocol that encourages research into and development of mainstream multi-user threshold signature wallet solutions.
- Providing modular and performant, free and open-source building blocks that allow software developers to either build new mainstream multi-user threshold signature wallets or integrate state-of-the-art threshold signature schemes into existing mainstream wallets.

6. References

[1] Canetti, R., Gennaro, R., Goldfeder, S., Makriyannis, N. and Peled, U. 2020. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security* (New York, NY, USA, 2020), 1769–1787. https://eprint.iacr.org/2021/060.

[2] Gennaro, R. and Goldfeder, S. 2020. One round threshold ECDSA with identifiable abort. Cryptology ePrint Archive, Paper 2020/540. https://eprint.iacr.org/2020/540.

[3] Canetti, R., Makriyannis, N. and Peled, U. 2020. UC non-interactive, proactive, threshold ECDSA. Cryptology ePrint Archive, Paper 2020/492. https://eprint.iacr.org/2020/492.

[4] Shamir, A. 1979. How to share a secret. *Commun. ACM*. 22, 11 (Nov. 1979), 612–613. DOI:https://doi.org/10.1145/359168.359176.

[5] Wikipedia. Polynomial interpolation: *https://en.wikipedia.org/wiki/Polynomial_interpolation*. Accessed: 2023-05-12.