<u>Whitepaper - MultiSub</u>

Table of contents

I) SUMMARY

II) CURRENT STATE OF WALLET MARKET

III) OUR SOLUTION: MultiSub

IV) TYPICAL USER PATH

V) TECHNICAL ARCHITECTURE

VI) BUSINESS MODEL

VIII) DRAWBACKS & NEXT STEPS

Project created in 36 hours by Dupin Alexandre, Haboury Pierre-Louis & Jouanny Léo.

I)Summary

MultiSub offers a curated multisig framework enabling day-to-day use with delegation of liquidity management while conserving custody.
The multisig remains the exclusive holder of all critical rights, but it can create sub-accounts with limited permissions: deposit-only, restricted withdrawals, interaction with a curated list of protocols, or execution of strictly controlled actions…

The goal is to simplify the user experience, which is currently fragmented across many hot wallets where users quickly get lost.
MultiSub addresses both retail users' needs to simplify the management of their different wallets/accounts and the requirements of institutional treasuries (DAOs, companies, funds, etc…).

II) CURRENT STATE OF WALLET MARKET

The wallet market is fragmented between retail-first smart wallets, multisig, and institutional custodians. On the retail side, Rabby has become the leading one for DeFi thanks to strong UX (transaction simulation, risk warnings, multi-chain support) and deep protocol integrations. But it still works like a classic hot EOA wallet: **security is tied to a single private key or seed, delegation is binary** (full control or nothing)**, and there is no native notion of granular sub-permissions or 24h spend limits.**

On the multisig side, **Safe** is the standard for crypto teams, DAOs, and treasuries. The multisigs greatly improve governance and reduce key risk, but remain inconvenient for day-to-day: workflows are heavy for small actions, and delegation usually means either full control of a given address.

Finally, **Coinbase Custody** and **Fireblocks** dominate the institutional end of the spectrum with regulated, insured custody and advanced operational tooling. They solve compliance and operational needs for funds, exchanges, and corporates, but at the cost of strong centralization and opaque risk models: users fully delegate key control to a third party, rely on proprietary infrastructure, and have limited composability.

Across all these models, the same gap remains: there is no simple way to combine multisig-grade security self-custody and programmability, granular delegation of permissions inside a single, unified wallet experience.

III) OUR SOLUTION: MultiSub

The Safe/Ledger multisig acts as your main bank account: it holds the funds and maintains full control. It can revoke any sub-account at any moment, adjust its limits, or add new authorisation.
Sub-accounts (e.g. mobile wallet, trading bot, yield manager, etc.) bring daily usability, but each operates strictly within the predefined scope.

Our sub-account system is based on two key components:

• **24h cumulative spending limits**

• **A strict whitelist of actions and protocols**.

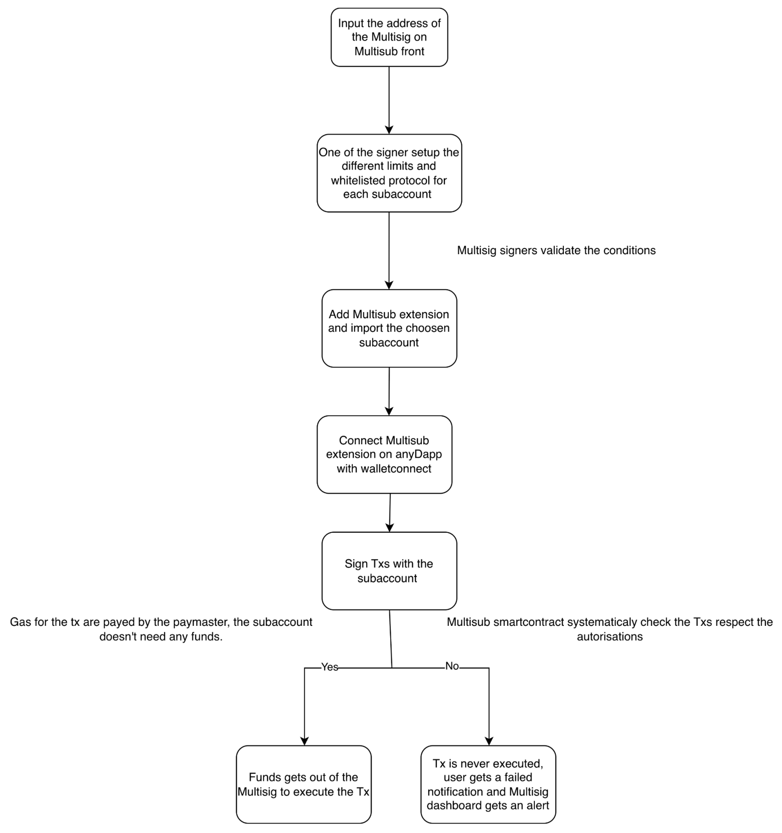
Together, these components create a secure, predictable, and modular execution environment without compromising usability/security.

This approach addresses the real needs of advanced users. For example, an individual can manage all their portfolio without ever exposing their entire assets; a family can share a common wallet with differentiated access; a DAO/fund/Tradfi entity can delegate strategy execution to a manager or an agent while retaining full control.

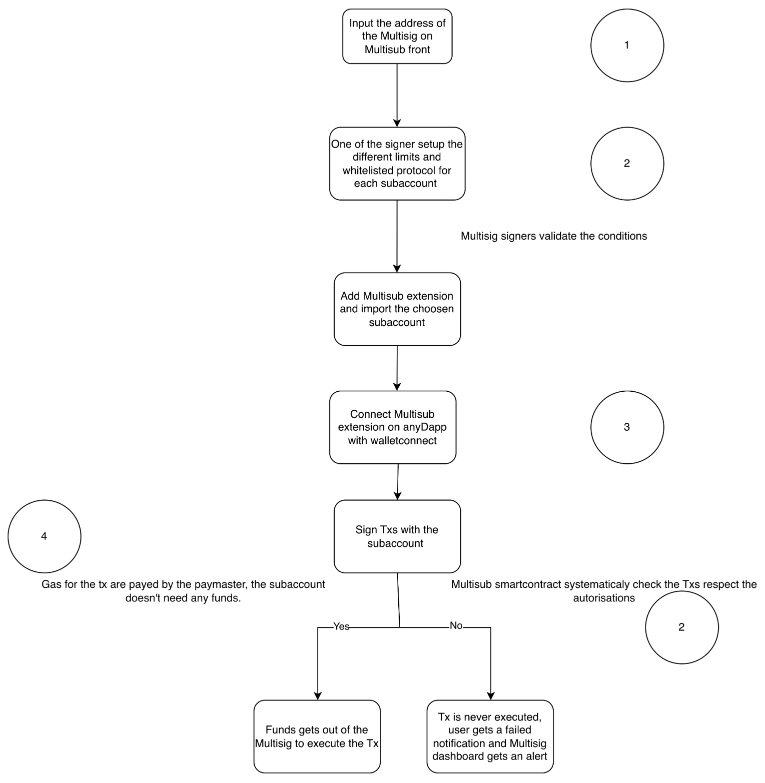
In short, we bring Morpho's vault advanced security principles into the world of collaborative custody.

The goal of our project is to combine three historically incompatible dimensions: **multisig-grade security**, **hot-wallet fluidity,** and **operational robustness**. To result in a new-generation self-custody model designed for users and institutions who need to operate efficiently without ever compromising on safety.

IV) Typical User Path


Input the address of the Multisig on Multisub front
One of the signer setup the different limits and whitelisted protocol for each subaccount
Multisig signers validate the conditions
Add Multisub extension and import the choosen subaccount
Connect Multisub extension on anyDapp with walletconnect
Sign Txs with the subaccount
Gas for the tx are payed by the paymaster, the subaccount doesn't need any funds.
Multisub smartcontract systematicaly check the Txs respect the autorisations
Yes
No
Funds gets out of the Multisig to execute the Tx
Tx is never executed, user gets a failed notification and Multisig dashboard gets an alert

V)SOFTWARE ARCHITECTURE


Input the address of the Multisig on Multisub front
1
One of the signer setup the different limits and whitelisted protocol for each subaccount
2
Multisig signers validate the conditions
Add Multisub extension and import the choosen subaccount
Connect Multisub extension on anyDapp with walletconnect
3
4
Sign Txs with the subaccount
Gas for the tx are payed by the paymaster, the subaccount doesn't need any funds.
Multisub smartcontract systematicaly check the Txs respect the autorisations
2
Yes
No
Funds gets out of the Multisig to execute the Tx
Tx is never executed, user gets a failed notification and Multisig dashboard gets an alert

1/ Safe/Ledger Multisig

Users input his multisig address on the multisub interface, which creates a dedicated guardian smart-contract (DeFiInteractorModule on the repo). The Multisig holds all funds and is the only entity allowed to enable/disable the module, configure roles, and trigger emergency actions. Technically, it acts as the Avatar/Owner pair in the Zodiac pattern, signing once to authorize the module and then only for governance-level changes, never for day-to-day DeFi interactions.

2/ Customization of the different authorization

One signer configures limits and protocol allowlists (needs to be signed by the threshold of the multisig); these parameters are stored in our custom Guardian contract. The contract embeds role management, 24h spend caps, and per-subaccount allowlists. On each transaction request coming from a subaccount, the contract checks if it complies with the configured rules before forwarding anything to the Safe. If Yes, the module calls the target DeFi protocol (Aave, Llamaswap etc...) and the Safe releases just enough funds to execute the operation. If No, the transaction is reverted, the user gets a failure notification, and the multisig dashboard surfaces an alert. This ensures that all protocol interactions are both composable and tightly constrained by the policies defined at the multisig level. This makes the contract the single policy engine that enforces guardrails while keeping logic isolated from the multisig itself.

3/ MultiSub extension directly interacts with the whitelisted Dapp

Each subaccount is a standard EOA (mobile wallet, bot, etc.) imported into the Multisub browser extension. These EOAs never talk directly to the Safe: they only call functions on the Guardian contract via the extension. It gives users a hot-wallet UX while still routing every operation through the contract to ensure it respects the guardrails.

4/ Account abstraction & Paymaster

When the user signs a transaction with a subaccount, it’s sponsored by the paymaster. The Guardian contract encodes the operation (call to the DeFiInteractorModule + target protocol), while the paymaster sponsors gas so fees can be paid by the multisig. The user gets a hot-wallet “click & sign” experience, while under the hood the module still enforces all limits and allowlists before any funds can move.

VI) BUSINESS MODEL

Just a cool tool for the community that we would love to use.

VII) DRAWBACKS & NEXT STEPS

Adding a guardian/module smart contract inevitably increases the attack surface: if the module is compromised, rules and limits could be bypassed. To mitigate this, the next steps are rigorous audits, formal verification of critical paths (limit checks, allowlists), and strict upgradability controls (timelocks, pause, multisig-gated upgrades) so that any bug can be patched without exposing user funds.

Users must install the Multisub extension to interact with dApps, which adds friction versus “just connect MetaMask/Rabby”. Our roadmap here is to minimize setup time (one-click onboarding, clear tutorials), support multiple browsers, and on the long term explore native integrations so Multisub can be surfaced inside existing wallets instead of always requiring a separate plugin.
Any change to limits, roles or allowlists must be validated by the multisig threshold, which can slow down operations. This is by design from a security standpoint, but we can soften the UX by adding scheduled updates, batched configuration changes, and preset templates tier-based on risk rating (currently emerging in DeFi with S&P arrival and Credora development by Redstone).

Supporting each protocol natively requires significant integration work. We partly solve this with a custom feature similar to adding a custom token in a wallet, but the long-term answer is to maintain a prioritized integration roadmap, open a simple adapter interface for partners, and progressively standardize on common patterns (ERC-4626, standard routers, etc.) to reduce per-integration cost.