# Master Thesis Project

Benjamin Strasser
13.03.2023

## Aufgabenstellung/Ziel

Das Ziel dieses Forschungsprojekt ist es, einen Prototyp einer Software Wallet zu erstellen, die einerseits dezentral und trustless ist, das heißt ohne zentrale Entität, die Zugriff auf den Private-Key hat, der vertraut werden muss und andererseits Usern eine intuitive Anwendung ermöglicht, die dem Erfahrungsbereich der Nutzer*innen entspricht. Auf Basis dieses Prototypens sollen grobe Designprobleme evaluiert und praktisch erprobt werden und als Basis einer weiterern Implementierung dienen.

Besonders relevant sind die aus dem Expose gezogenen Fragestellungen, welche für das Projekt angepasst wurden.

• Inwiefern kann eine Wallet mit adäquater Security und Dependability entwickelt werden, die in Hinblick auf die Nutzererfahrung dem Web 2.0 Nahekommt?

• Wie kann eine praktische Umsetzung einer Wallet erreicht werden, die ein Login beinhaltet, das jener einer standardisierten Website ähnelt und welche Implikationen in Bezug auf den Kriterienkatalog ergeben sich daraus? Speziell ist die Frage ob eine Web Wallet technisch umsetzbar ist?

• Wie kann das Signieren der Transaktion so programmiert werden, dass es jenem des E-Bankings mit Security App nahekommt?

• Wie ist es möglich, Interaktionen mit einem Smart-Contract sicher und user-freundlich abzuwickeln.

## Projektbeschreibung und Outcome

Das Projekt wurde als Monorepo mit Turborepo aufgesetzt. Dabei gliedert es sich in 4 Teilprojekete. Drei Sveltekit Anwendungen und eine Hardhat Anwendung.

Die Hardhat Anwednung stellt eine lokale Ethereum Node zur Verfügung und einen Dummy Testcontract. Außerdem werden Deploymentadressen und Typechain Types generiert, welche in allen anderen Anwendungen verwendet werden können.

Die Wallet Client ist eine Testanwendung in Sveltekit, welche eine EIP-1193 konforme Wallet implementiert. Hierbei wurde erfolgreich die Kommunikation mit der Webwallet getestet und festgestellt, dass der grundästliche Gedanke über EIP-1193 als Standard für die Signatur und Abfrage von Informationen zu verwenden funktioniert. Sämtliche Blockchain Informationen und Transaktionen können abgewickelt werden. Der Versuch Whisper oder andere Technologien zur Kommunikation mit der Webwallet einzubinden ist jedoch nicht geglückt. Sowohl Hardhat Libraries haben hier nicht funktioniert als auch lokal gestartete geth Docker Container. Für die Kommunkation wurde daher einfach ein zentraler Server verwendet der Messages annimmt und weiterleitet. Auch wurde keine Implementierung eines dezentralen Usermanagmenents umgsetzt, da sich ein einfacher Login mit Passwort(nicht SSO via Google etc.) als trivial genug dargestellt hat und daher im Prototypen nicht relevant ist.

Die Webanwendung hat die Implementierung des Shamir-Secret Sharing Algorithmus plausibel gemacht. Dabei wurde gezeigt, dass es möglich ist eine eventuell verschlüsselte Mnemonic an drei Nodes zu verteilen und und nach erfolgreichem recovery von zwei der Shards die Mnemonc wiederherzustellen. Die Nodes wurden einfachheitshalber gemocked, da die Implementierung eines einfachen Key Value Store mit Basic User Authentication trivial erschien. Das Usermanagement wurde wie bei dem Wallet Client komplett ignoriert und angenommen, dass es nur einen User gibt. Die Key Shards wurden einfach in-memory gespeichert.

Der Node Mock und die Kommunikation wurde in der dritten SvelteKit Anwendung implementiert.

# Lessons Learned

Das grundsätzliche Design einer Webwallet, wie ich Sie im Laufe des Projekts konzipiert habe sollte möglich sein. Die Kommunikation zwischen Client und Wallet könnte sich noch als problematisch herausstellen, da die meisten dezentralen Kommunikationstechnologien nicht ausgerteift sind. Shardstorage ist bei Anbietern wie Google etc. leider nicht möglich und ein klassicher Username Password vorgang im Zusammenhang mit dezentralen Nodes scheint am sinnvollsten. Dadurch sollte es möglich sein einen dezentralen Web-Login für User zu ermöglichen und via Username und Anti-Spam Code Anfragen aus DApps an die Anwendung zu schicken.

# Aufwandsaufstellung

| Aufgabe | Aufwand in Stunden |
|---|---|
| Paper Wallet- /Architekturdesigns | 52 |
| Meetings/Presentation Vorbereitung | 8 |
| Aufsetzen Turborepo Sveltekit | 4 |
| Anbindung SSO/ Datenspeicher Microsoft Google/Github | 13 |
| Libarry Evaluering Shamir Secret Sharing | 8 |
| Whisper Hardhat Geth/ Evaluierung Alternativen | 26 |
| Entwicklung/Research EIP-1193 konformer Wallet | 18 |
| Anbindung EIP-1193 Wallet Client und Webwallet | 6 |
| Node Mock KeyShards Registrierung und Load | 16 |
| Kommunikation Wallet Client Webclient | 12 |
| Bericht | 2 |

Link Github: https://github.com/benjaminstrasser/Master-Project

Das Video und PDF befinden sich ebenfalls im Github.