



Abbildung 1: Titelbild

# Decentralized Key Recovery

Key Recovery für DKMS-basierte Systeme

## Projekt 2

Studiengang:	Bachelor of Science
Autor:	Dario Furigo, Beat Schärz
Betreuer:	Gerhard Hassenstein, Annett Laube
Auftraggeber:	Gerhard Hassenstein, Annett Laube
Experten:	Gerhard Hassenstein, Annett Laube
Datum:	12.06.2020

## Management Summary

Die aufkommende Self-Sovereign Identity (SSI) Technologie erfordert eine praktikable, sichere und robuste Lösung des Key Recovery, welche einfach und intuitiv zu bedienen ist. Diese Arbeit befasst sich mit den verschiedenen Key Recovery Ansätzen im Allgemeinen und analysiert die momentane Lage der bestehenden Applikationen und deren Funktionalitäten. Ausserdem beleuchtet sie offene Punkte und Herausforderungen, welche bei den jeweiligen Lösungen noch nicht überwunden sind.

Die Arbeit besteht einerseits aus einer Abhandlung der zur Verfügung stehenden Möglichkeiten eine Key Recovery Lösung umzusetzen und andererseits aus den dazu gefunden Applikationen. Es fliessen zudem Ideen und Gedanken ein, wie bestehende Lösungen ergänzt und verbessert werden können. Weiter sind Kriterien festgelegt, um die Applikationen nach ihren Fähigkeiten und ihrer Anwendbarkeit für den SSI Use Case zu bewerten.

Die Resultate zeigen auf, welche Key Recovery Technologien am besten geeignet sind und welche Anwendungen diese am besten umsetzen. Leider gibt es keine Applikation, welche allen Kriterien gerecht wird. Trotzdem konnte im Kapitel "Showcase" mit einer Applikation ein tatsächliches Backup und Recovery eines kryptografischen Schlüssels durchgeführt und dokumentiert werden.

Abschliessend lässt sich sagen, dass das Key Recovery für den SSI Use Case für den Moment ungelöst bleibt. Es gibt aber gute Ansätze, wovon bereits einige in Applikationen eingebaut sind, die jedoch meist im Gebiet der Kryptowährungen zu Hause sind. Solange SSI an Relevanz gewinnt, ist eine Lösung des Key-Recovery-Problem für SSI nur noch eine Frage der Zeit.

# Inhaltsverzeichnis

1	Einleitung	4
2	Self-Sovereign Identity	5
2.1	Die drei Modelle digitaler Identitätsbeziehungen	5
2.2	Die ungelöste DKMS Frage	6
2.3	Mögliche Lösung	6
3	Key Recovery Technologien	8
3.1	Mnemonic / Seed Phrase	8
3.2	Social Recovery	8
3.3	Threshold Signatures / MultiSig	8
3.4	Key Escrow	9
3.5	Biometrics	9
3.6	Fazit Key Recovery Technologien	9
3.7	Fragen und Probleme von Social Recovery	10
4	Kriterienkatalog	12
5	Applikationen	14
5.1	Connect.me	14
5.2	uPort	14
5.3	Vault12	15
5.4	Argent	15
5.5	Dark Crystal	16
5.6	ZenGo	17
5.7	Bewertung	18
5.8	Fazit	19
6	“Showcase”	20
7	Schlussfolgerungen	22
8	Diskussion	23
9	Abbildungsverzeichnis	24
10	Tabellenverzeichnis	24
11	Glossar	25
12	Literaturverzeichnis	26
13	Anhang	28
13.1	Showcase	28
14	Selbständigkeitserklärung	32

# 1 Einleitung

Seit es Passwörter gibt, existiert auch die Frage "Was, wenn man sein Passwort verliert?". Deswegen ist jeder mit den konventionellen Prozessen des Passwort-Recovery vertraut: Man wählt die Passwort-Vergessen-Funktion aus, gibt seine E-Mail ein und erhält entweder einen Link für den Passwort-Wechsel oder direkt ein neues, temporäres Passwort. Der Prozess ist einfach, verständlich, relativ simpel umsetzbar und grossflächig etabliert. Jedoch ist allgemein bekannt, dass ein gutes Passwort viele Bedingungen erfüllen muss und es nicht selten irgendwo notiert und/oder wiederverwendet wird. Erhöht man die Sicherheit mit zusätzlichen Faktoren, geschieht dies oftmals auf Kosten der User Experience.

Eine sichere Alternative zu Passwörtern bieten asymmetrische Kryptografische Schlüssels (Keys), welche insbesondere dank ihrer hohen Entropie, die es einem Angreifer beinahe unmöglich macht, sich durch Brute-Force Zugriff zu verschaffen, das bevorzugte Mittel für vielerlei Anwendungen sind. Das typische Beispiel für diese Anwendungen sind Kryptowährungen wie Bitcoin. Verliert man dort den Private Key zu seinem Wallet, sind jegliche Coins nicht mehr zugänglich. Ein weiterer Use Case, welcher als Ausgangslage dieser Arbeit diene, ist "Self-Sovereign Identity (SSI)". Hierbei werden Identitätsbeziehungen mit Hilfe von Private- und Public Key Paaren erstellt. Verliert man dort sein Wallet und hat keinen Weg es zu restoren, sind alle Identitätsbeziehungen verloren. Stellt man sich hier nun also dieselbe Frage "Was, wenn man seinen Key verliert?", stellt man schnell fest, dass das Problem nicht so einfach mit einem Passwort-Reset lösbar ist.

Im Normalfall und in den Fällen, welche in dieser Arbeit behandelt werden, liegt der Private Key einzig und allein beim entsprechenden User und ein allfälliges Gegenüber besitzt nur den Public Key, um den User zu verifizieren. Es gibt also keine Person, welche den Key, z.B. im Falle eines kaputten Smartphones, zurücksetzen oder wiederherstellen kann. Weiter stellt sich die Frage, was bei einem gestohlenen Private Key passiert. Gibt es eine Möglichkeit diesen zu revozieren oder auszutauschen?

Zusätzlich spielen andere Faktoren eine Rolle: Will man einen verlorenen Key, welcher für Signaturen verwendet wurde, überhaupt wiederherstellen? Man weiss nicht, ob ihn jemand anderes entwendet hat und nun auch benutzt. Falls man aber Daten entschlüsseln muss, kommt man jedoch nicht um eine Wiederherstellung herum.

Die Lösungen für das Wiederherstellen eines Private Keys sind vielseitig. Der am meisten verbreitete Ansatz ist eine Seed Phrase (auch Mnemonic genannt), mit welcher man aus einer Reihe von Wörtern deterministisch einen Private Key wiederherstellen kann.

Eine andere Methode ist das Social Recovery, mit welchem der Key auf verschiedene vertrauenswürdige Entitäten aufgeteilt und bei Bedarf wiederhergestellt werden kann.

Auch weit verbreitet ist ein zentrales Backup bei einem Provider. Dies ist jedoch nicht im Sinn dieser Arbeit, da eine dezentrale Lösung gesucht werden soll. Dabei ist das Stichwort DKMS von zentraler Bedeutung.

Diese und andere Methoden, die darauf basierenden Anwendungen und das Problem als Ganzes soll in dieser Arbeit von verschiedenen Seiten beleuchtet werden.

Das Ziel ist es, eine Lösung zu finden, welche für SSI nutzbar und dezentral gehalten ist.

## 2 Self-Sovereign Identity

Gemäss einem Artikel von Evernym (Ruff 2018), einem der Vorreiter der Technologie, ist SSI wie folgt erklärbar:

Generell besteht im Identitätsmanagement, als auch mit anderen Technologien, das Paradox, dass Security nur auf Kosten der User Experience verbessert werden kann und umgekehrt.

Bei SSI handelt es sich um eine vielversprechende Technologie, welche das Identitätsmanagement im Internet revolutionieren und sowohl die User-Experience als auch die Security Aspekte wesentlich verbessern soll.

SSI galt als Grundlage dieser Arbeit, da hier das Key-Recovery-Problem weitgehend noch ungelöst ist.

### 2.1 Die drei Modelle digitaler Identitätsbeziehungen

SSI ist am verständlichsten, wenn es mit den herkömmlichen Varianten des Identitätsmanagements verglichen wird, weshalb diese hier kurz beschrieben werden.

#### Silos / traditionelle Modelle

Silos sind die traditionelle und simpelste Art von Identity Management. Das Vertrauen zwischen dem User und der Organisation basiert auf Secrets, im klassischen Fall: Username und Passwort. Unter Umständen werden zusätzliche Faktoren wie PINs, physikalische Tokens etc. verwendet. Gespeichert sind alle Daten auf den Servern der Organisation.



Abbildung 2: Traditionelle Identität

#### Third Party IDP (IDentity Provider)

In diesem Modell verlässt man sich auf eine dritte Organisation, welche das Identitätsmanagement übernimmt und die Identität des Users an den entsprechenden Service "fördert".

Dabei sind Protokolle wie OpenID Connect oder SAML im Einsatz. Die Beziehung zwischen Enduser und IDP ist dabei gleich gesichert wie beim traditionellen Modell. Ein typisches Beispiel für dieses Modell sind soziale Logins wie "Facebook Login".



Abbildung 3: Identity Provider

#### Self-Sovereign / Peer-to-Peer

Mit dem SSI Modell gibt es keinen Dritten im Identifizierungsprozess mehr.

Jeder Teilnehmer besitzt ein digitales Wallet, in welchem "Verifiable Credentials" enthalten sind. Diese beinhalten Informationen darüber, wer sie ausgestellt hat, an wen sie ausgestellt wurden, ob sie verändert wurden und ob sie revoziert wurden. Diese Credentials können von jedem Teilnehmer des SSI Netzwerks ausgestellt werden. Gewisse Credentials werden dabei öffentlich auf einer Blockchain referenziert, damit deren Inhalt von einer Dritten Partei geprüft werden kann.

Weiter werden im Wallet alle Beziehungen des Teilnehmers gepflegt, welche dieser durch Austauschen von "Identifiers" mit dem entsprechenden Peer, erstellen kann. So entsteht ein Vertrauen, bei welchem man sich auf keine Dritten verlassen muss.

Die ultimative Vision von SSI wäre, dass jede Person, Organisation und jedes "Thing" (Internet of Things) jegliche Art von Credentials an irgendeine andere Person, Organisation oder ein anderes "Thing" ausstellen kann, welches wiederum von jeder teilnehmenden Partei verifiziert werden kann.

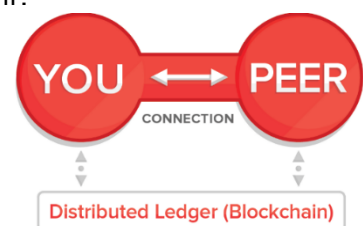


Abbildung 4: SSI

## Gegenüberstellung

	Silo / traditionell	Third Party IDP	SSI
Userfreundlichkeit	<ul style="list-style-type: none"> <li>Einfach zu verstehen</li> <li>Hunderte Credentials -&gt; vergessene Passwörter</li> </ul>	<ul style="list-style-type: none"> <li>Nur ein Credential</li> </ul>	<ul style="list-style-type: none"> <li>Vision: sehr gut / one-click</li> </ul>
Sicherheit	<ul style="list-style-type: none"> <li>Wenn Passwörter gut sind und nicht wiederverwendet werden -&gt; sehr sicher.</li> <li>Wiederverwendete Passwörter</li> <li>Einbrüche in ein System legen ggf. Persönliche Daten und Credentials offen</li> <li>Phishing möglich</li> </ul>	<ul style="list-style-type: none"> <li>Nur ein Credential welches sicher gehalten werden muss</li> <li>Phishing möglich</li> </ul>	<ul style="list-style-type: none"> <li>Keine Passwörter, sondern kryptographische Keys</li> <li>Kein Phishing (gegenseitige Authentifizierung)</li> <li>Privater Kommunikationskanal (End-to-end Verschlüsselung)</li> </ul>
Privatsphäre	<ul style="list-style-type: none"> <li>Accounts sind per Username oder Mail assoziierbar</li> </ul>	<ul style="list-style-type: none"> <li>Zentraler Identity Provider kann viele Daten sammeln</li> </ul>	<ul style="list-style-type: none"> <li>Anonymität (keine Assoziierbarkeit)</li> <li>Mit Zero-Knowledge-Proofs können nach Wunsch auch nur Teile eines Credential offengelegt werden</li> </ul>
Einschränkungen	<ul style="list-style-type: none"> <li>Keine Abhängigkeit von Dritten</li> </ul>	<ul style="list-style-type: none"> <li>Login abhängig von Dritten</li> </ul>	<ul style="list-style-type: none"> <li>Wechsel zu neuer Lösung generiert Kosten</li> <li>Key Recovery noch ungelöst.</li> </ul>

Tabelle 1: Gegenüberstellung SSI

## 2.2 Die ungelöste DKMS Frage

Wenn man bei einer traditionellen Webseite sein Passwort vergisst, gibt es immer noch die Möglichkeit die "Passwort vergessen"-Funktion der Webseite einzusetzen.

Bei SSI verhält sich das aber wie bei Bitcoin: Verliert man seine Keys, verliert man seine Coins bzw. im Fall von SSI die digitale Identität. Das Management und insbesondere das Recovery der hunderten von Keys, welche man in seinem Leben erstellen und in seinem Wallet ablegen wird, könnte zur Achillesferse von SSI werden.

Folgende und noch viele Fragen mehr stellen sich:

- Wie werden Keys gesichert?
- Wohin werden diese Backups geschrieben?
- Wie kann gegebenenfalls ein Master-Key eingesetzt und im Falle eines Recovery wiederhergestellt werden?
- Wie muss dieser Prozess gestaltet werden, damit er auch von Personen ohne technischen Background navigiert werden kann?

Solange diese Fragen nicht geklärt sind, wird die breite Bevölkerung die SSI Lösungen nicht einsetzen können.

## 2.3 Mögliche Lösung

Eine mögliche Lösung für das Sichern eines SSI Wallet kann man sich so vorstellen:

Das Wallet wird als Ganzes entweder mit einem symmetrischen Key oder einem Public Key verschlüsselt. Der symmetrische Schlüssel oder der passende Private Key dient nun als Master-Key. Somit sind mit Hilfe eines einzigen Keys alle Schlüssel des Wallets gesichert.

Damit das Recovery gewährleistet werden kann müssen zwei Dinge sichergestellt werden:

- Der Master-Key muss so gesichert werden, dass er bei Bedarf innerhalb angemessener Zeit durch den Besitzer zurückgeholt werden kann. Ausserdem soll der Key zu jeder Zeit nur durch den Besitzer abrufbar sein.
- Das verschlüsselte Wallet muss so gesichert werden, dass es bei Bedarf innerhalb angemessener Zeit erreicht werden kann. Da das Wallet verschlüsselt ist, kann dieses z.B. auch "in der Cloud" oder in einem dezentralen Storage-System aufbewahrt werden, ohne die Sicherheit der gespeicherten Schlüssel zu gefährden.

## 3 Key Recovery Technologien

### 3.1 Mnemonic / Seed Phrase

Ein Mnemonic ist eine bestimmte Anzahl natürlicher Wörter (genannt Seed Phrase), mit welchem ein Private Key deterministisch abgeleitet werden kann. Es gibt eine kleine Anzahl Standards namens Bitcoin Improvement Proposals BIP (Bitcoin 2020), welche beschreiben wie viele Wörter ein Mnemonic beinhaltet und wie der Seed daraus abgeleitet werden kann. Meist werden die Standards BIP32, BIP39 oder BIP44 verwendet.

Der User muss diese 12-24 Wörter niederschreiben und sicher aufbewahren. Die Methoden der Aufbewahrung gehen dabei von handgeschriebenen Zetteln, aufbewahrt im Nachttisch oder in einem Banktresor, über Gravuren in Metall, um die Wörter feuer- und wasserfest zu machen, bis hin zu "Brainwallets", welche das Konzept des Auswendiglernens dieser Phrases beschreiben.

Bei all diesen Vorgängen liegt die Verantwortung voll und ganz auf der Phrase und darauf, dass der User diese sicher aufbewahren kann. Die Lösung ist weder benutzerfreundlich, noch kann man sich darauf verlassen, dass jeder User die Dringlichkeit einer sicheren Ablegung versteht.

Die Lösung ist aber nicht nur schlecht. Wenn der User eine sichere Aufbewahrung umsetzt, ist der Recovery Prozess relativ schnell, unabhängig von Dritten und natürlich auch sicher. (Pagliari 2019)

### 3.2 Social Recovery

Hinter Social Recovery steckt die Idee, dass der User, welcher seinen Key sichern möchte, diesen in mehrere Teile ("Shares") aufsplittet und die Einzelstücke dann an vertraute Entitäten verteilt.

Bei den vertrauten Entitäten kann es sich um Organisationen wie Banken oder Versicherungen, um die Regierung, aber insbesondere auch um Bekannte oder Verwandte handeln, welchen man die Aufbewahrung eines Shares anvertraut.

Da die anderen Parteien ihrerseits auch ihren Storage oder den Zugang dazu verlieren können, kann hier auf ein "M von N"-Schema gesetzt werden. Falls gewünscht, ist es somit möglich, dass nicht alle Shares benötigt werden, sondern z.B. nur 3 von 5.

### Shamir Secret Sharing

Der bekannteste Algorithmus, um ein Secret in mehrere Shares aufzuteilen ist dabei das "Shamir's Secret Sharing"-Schema, welches schon 1979 in einem Paper mit dem Namen "How to share a secret" von Adi Shamir vorgestellt wurde (Shamir 1979). Die Grundlage dabei bieten polynomiale Gleichungen. Es werden  $k$  Punkte benötigt um ein Polynom des Grades  $k-1$  festzulegen. Beim Schema wird nun zuerst das Secret auf der Y-Achse definiert:  $P(0, \text{"Secret"})$  und danach ein zufälliges Polynom mit dem gewünschten Grad generiert. Nun können so viele Punkte auf diesem Polynom als Share ausgewählt werden wie gewünscht sind (Anzahl Personen, welche einen Share erhalten sollen). Egal wie viele Punkte man dann rausgibt, es braucht immer genau  $k$  Punkte, um das Polynom zeichnen zu können, somit den Schnittpunkt mit der Y-Achse zu erhalten und das Secret wiederherzustellen.

### 3.3 Threshold Signatures / MultiSig

Threshold Signatures finden ihren Einsatz primär bei Transaktionen auf der Blockchain, welche aber von einer bestimmten Menge von Teilnehmern bestätigt werden muss. Generell berechnen hier die Teilnehmer zusammen einen Public Key und halten jeweils einen eigenen geheimen Share des Private Key. Die Signatur einer Transaktion wird nun von "M von N" Teilnehmern signiert, bevor sie gültig ist. So können  $N$  minus  $M$  private Teile verloren gehen und die Transaktionen können trotzdem noch durchgeführt werden. Wenn die Blockchain das TSS (Threshold Signature Scheme) als Teil ihrer Funktionalität unterstützt spricht man von "Multisig". Dabei werden mehrere Private Keys für die Signatur verwendet. (Shlomovits, Threshold Signatures Explained kein Datum) Für den Use Case dieser Arbeit sind TSSs nicht direkt einsetzbar, trotzdem werden sie hier erstens der Vollständigkeit halber aufgelistet und zweitens gibt es folgenden Ansatz wie es verwendet werden könnte:

Der User, der seine Keys sichern möchte, legt öffentlich den gemeinsamen Public Key einer Gruppe von Personen fest, welcher er vertraut. Verliert nun der User seinen Private Key, erstellt er sich ein neues Key Pair und meldet sich bei  $N$  von  $M$  Personen der Gruppe und gibt ihnen den neuen Public Key. Nachdem sie überprüft haben ob es sich um den erwarteten User handelt, threshold-signieren sie



diesen und veröffentlichen eine Ankündigung, dass der neue Key nun gültig sei und der alte ist somit auch gleich revoziert. (Sean Gilligan 2019)

### 3.4 Key Escrow

Key Escrow beschreibt eine Vorkehrung, bei welchem der asymmetrische oder symmetrische Key unter Verschluss gehalten wird, bis der User oder ein Dritter diesen anfordert und gewisse Bedingungen erfüllt. Key Escrow wird meist im Zusammenhang mit staatlichen Unternehmen genannt, bei welchem Privatpersonen oder Firmen gezwungen werden Keys zu hinterlegen, um dem Staat, falls nötig, Zugriff darauf zu ermöglichen. Der Service kann aber auch firmenintern umgesetzt werden, damit im Falle eines Verlusts, der User wieder zu seinem Key kommt oder nach dem Austritt eines Users die Rechtsabteilung Zugriff auf seine verschlüsselten Daten bekommt.

Es gibt Firmen, welche sich auf Key Escrow Services spezialisiert haben. Der Escrow-Anbieter ist hier natürlich ein weiteres Sicherheitsrisiko, da es sich gegebenenfalls um einen Dritten handelt, welcher so Zugriff auf den Private Key bekommt.

### 3.5 Biometrics

Biometrische Attribute können auf drei verschiedene Wege eingesetzt werden, um die Sicherheit von Private Keys und deren Recovery zu gewährleisten.

Der erste ist biometrische Attribute in die Generierung der Keys einfließen zu lassen.

Der zweite wäre eine biometrische Authentifizierung vor dem Öffnen des Keys auf dem Gerät, welche den Private Key speichert.

Die dritte verschlüsselt direkt die Private Keys mit biometrischer Sicherheit. (Mehmet Aydar 2019)

Biometrics bieten den riesigen Vorteil, dass sie (in den allermeisten Fällen) nicht verloren gehen und man sie immer auf sich trägt. Weiter können sie mit anderen Methoden und Faktoren kombiniert werden, was die Sicherheit noch weiter erhöht.

Das grosse Problem bei den Biometrics ist, dass sich die Revozierung schwierig umsetzen lässt, da man sich z.B. nicht einfach einen neuen Finger machen kann, sobald ein Fingerprint gestohlen wurde.

Noch schwieriger verhält sich das mit einmaligen Eigenschaften wie dem Gesicht, der Stimme etc.

Auch wenn der Fall unwahrscheinlich ist, darf ausserdem ein "zerstörer" Finger das Recovery nicht unmöglich machen.

### 3.6 Fazit Key Recovery Technologien

Mnemonic ist auf jeden Fall eine zeitgetestete und weitverbreitete Lösung. Sie ist einfach umzusetzen und auch nicht allzu schwer verständlich. Das grosse Problem liegt hier dabei, dass sich ein User bewusst sein muss, wie unentbehrlich ein Backup der Phrase ist.

Key Escrow funktioniert natürlich, ist aber in den allermeisten Fällen nicht im Sinne des Key-Besitzers, da so ein Dritter vollen Zugriff auf den Key hat. Insbesondere für einen Use Case wie SSI wäre ein Anvertrauen des Keys an einen Dritten unvorstellbar.

Biometrics scheinen auf den ersten Blick eine perfekte Lösung zu sein. Sie können nicht verloren gehen und man trägt sie immer bei sich. Allerdings ist die deterministische Ableitung von Private Keys aus biometrischen Attributen noch nicht so weit wie sie sein müsste. Somit bleibt die Möglichkeit eines Safes oder einer Software, welche/r z.B. mit einem Iris-Scan oder einem Fingerprint geöffnet würde. Allerdings gab es bereits Angreifer, welche aus einem Bild in den Sozialen Medien den Fingerprint eines Individuums stehlen konnten. Dies wird wohl in der Zukunft nur noch einfacher. Kombiniert mit der Revozierungsproblematik sind auch Biometrics keine perfekte Lösung.

Bleibt nur noch Threshold Signatures und Social Recovery. Beide Ansätze verfolgen ein sehr ähnliches Prinzip: Man verteilt Keys auf vertraute Entitäten wie Freunde, Bekannte oder Organisationen und erhofft sich, dass mindestens ein Teil dieser Entitäten für den Recovery Fall zur Verfügung stehen und ihren Share einsetzen könnten.

Der grosse Vorteil von Social Recovery liegt darin, dass kein öffentlicher Ledger oder ähnliches zur Verfügung stehen muss, währenddessen eine Anfrage irgendwo veröffentlicht werden müsste, damit sie mit TSS signiert werden kann. Social Recovery setzt meist auf eine Peer-to-Peer Lösung, welche unabhängig von zentralen Applikationen und Speicherlösungen arbeitet. Distributed Ledger haben

zudem die Schwäche, dass sie durch grosse Rechenkapazität kontrolliert werden können. Aus den genannten Gründen ist Social Recovery die bevorzugte Lösung.

### 3.7 Fragen und Probleme von Social Recovery

Aus dem vorangehenden Kapitel geht hervor, dass Social Recovery die geeignetste Lösung für das Problem zu sein scheint. Allerdings gibt es bei diesen Ansätzen auch gewisse Nachteile, welche einem bewusst sein müssen und gewisse Fragen, um welche man sich kümmern muss.

#### Nachteile

Wie aus der Definition ersichtlich ist, ist es möglich, dass die vertrauten Entitäten einen Collusion-Angriff (dt. "Absprache") gegen den Besitzer des geteilten Secrets starten. Dabei müssen sich M der N Teilnehmer absprechen und können so den Key ohne das Wissen des Besitzers wiederherstellen. Deswegen ist es wichtig, dass die vertrauten Entitäten mit diesem Gedanken gewählt werden und die Entitäten anonym bleiben.

#### Gedanken und Fragen

Rund um die ganze Social Recovery Thematik gibt es viele Fragen, die sich vor einer Implementation stellen. Hier eine unvollständige Liste dieser Punkte:

- "Circles"

Alice möchte ein "3 von 9" Schema, wobei sie 3 Schlüssel jeweils an Freunde, an Verwandte und an Business-Partner verteilt. Allerdings möchte sie nicht, dass die Business-Partner allein den Restore durchführen können, obwohl sie zusammen 3 Shares haben. Dafür kann das Konzept von "Circles" (Kreise / Gruppen) eingeführt werden, mit welchen Alice bestimmen kann, dass z.B. von jeder Gruppe (Freunde, Verwandte, Business-Partner) ein Share benötigt ist. (Christopher Allen 2019)

- Verifizierbare Shares

Wenn bei einem Wiederherstellungsversuch der generierte Key nicht stimmt, gibt es für den Besitzer des Keys keine Möglichkeit herauszufinden, welcher der Teilnehmer einen falschen Share eingereicht hat. Diese Problematik kann z.B. mit "Verifiable Schemes" gelöst werden. Zudem gibt es auch für die Teilnehmenden Möglichkeiten herauszufinden, ob der Besitzer ("Dealer") auch wirklich gültige Shares herausgegeben hat. (Peg 2019)

Zu den gängigsten Schemen zählen: Feldmann's Scheme (Feldman 1987), Pederson Scheme (Pedersen 1991) und Schoenmaker's Scheme (Schoenmakers 1999).

- Verlorener Trust / Entfernen einer vertrauten Entität

Verliert man das Vertrauen in einen oder mehrere Teilnehmer, muss es möglich sein, diesen aus dem Prozedere auszuschliessen. Beim Schema von Shamir können theoretisch unendlich viele Mengen von Shares für das gleiche Secret erstellt werden. So könnte beim Vertrauensverlust eine neue Menge erstellt und diese nur noch an die neu vertrauten Parteien verteilt werden. (Peg 2019)

- Motivation der Teilnehmer

Die Teilnehmer welche Shares aufbewahren, müssen einen Anreiz haben dies zu tun. Bei Freunden und Verwandten kann man sich hier wahrscheinlich auf deren Mithilfe aus Wohlwollen verlassen, anders sieht das eventuell bei Organisationen aus. Diese müssen eine Motivation besitzen den Share korrekt aufzubewahren. (Sean Gilligan 2019)

- Single Point of Compromise

Das Gerät, welches den Key aus den Shares zusammenstellt, stellt ein Security Risiko dar. Folgende Fragen müssen geklärt werden: Wo wird die Prozedur durchgeführt? Bleibt danach etwas im Speicher des Geräts? Wo wird der resultierende Key gespeichert? (Sean Gilligan 2019)

- Anonymität

Anonymität innerhalb der Teilnehmer kann gewünscht sein, da es für diese so massiv schwieriger wird einen Collusion-Angriff zu starten. Allerdings zieht das mit sich, dass es schwierig oder unmöglich wird, bei einem Todesfall des Besitzers den Key wiederherzustellen. (Sean Gilligan 2019)

- "Health Check"

Die Verteilung der Shares bringt nur solange etwas, wie der Share auch tatsächlich zur Verfügung steht. Wenn fünf Shares verteilt werden, aber im Verlauf der Zeit drei verloren gehen, dann ist der Aufwand vergebens gewesen. Deswegen kann es sinnvoll sein, dass der "Dealer" regelmässig überprüfen kann, ob die Shares noch vorhanden sind. (Sean Gilligan 2019)

- Rechtlicher Aspekt

Was passiert in einem Todesfall des Besitzers? Wenn es sich zum Beispiel um Kryptowährungen handelt und diese vererbt werden sollen. Wer stellt diese wieder her?

- Welche Daten werden geteilt?

Welche Art von Daten müssen wiederhergestellt werden? Sind es die Secrets (Passwörter, Private Keys usw.) oder macht es auch Sinn andere Daten wie zum Beispiel verschlüsselte Wallets oder wichtige Dokumente zu teilen, um diese wiederherzustellen.

## 4 Kriterienkatalog

Die gesuchte Applikation, sollte nachstehende Kriterien erfüllen bzw. diese zumindest auf der Roadmap haben.

### K1: Benutzerfreundlichkeit

Die wenigsten bekannten Lösungen für Key Recovery sind einfach zu handhaben. Deshalb ist es ein wichtiges Kriterium, dass die Applikation auch durch normale Alltagsanwender und nicht nur durch Spezialisten bedienbar ist. Insbesondere bei SSI als Use Case ist davon auszugehen, dass dies in absehbarer Zukunft für alle Menschen und nicht nur für technisch versierte Anwender eine Rolle spielen kann und auch soll.

Dazu wird benötigt:

- Eine Anzeige, ob das Backup erfolgt ist bzw. noch eingerichtet werden muss.
- Eine Anzeige, ob das Backup noch aktuell bzw. geschützt ist.
- Einen Recovery-Prozess, welcher intuitiv und ohne grösseren Aufwand navigierbar ist.

### K2: Verfügbarkeit für alle gängigen Plattformen

Die Applikation sollte für alle gängigen Plattformen erhältlich sein. Somit sollten mobile Betriebssysteme wie iOS und Android als auch nicht-mobile Betriebssysteme wie Linux, MacOS und Windows unterstützt werden.

### K3: OpenSource

Der Code der Applikation sollte frei einsehbar und verifizierbar sein. Zudem soll eine Prüfung der Funktionen und eine Erweiterung der Funktionalität einfach möglich sein.

Falls wir keine Applikation finden, die allen unseren Anforderungen gerecht wird, müssten wir für die Bachelorthesis gegebenenfalls die Applikation anpassen. Somit wird vorausgesetzt, dass eine Lizenz verwendet wird, bei welcher die Applikation uneingeschränkt angepasst werden kann und auch wieder als OpenSource Software veröffentlicht werden darf.

### K4: Dokumentation

Die Dokumentation sollte sich nicht nur auf ein Whitepaper, welches als Marketingmaterial fungiert, beschränken. Stattdessen sollten jegliche Funktionen umfangreich beschrieben und nachvollziehbar sein.

### K5: Key Recovery

Das Key Recovery sollte dezentralisiert gelöst werden. Dies ist ein Muss-Kriterium und eine Grundlage für diese Arbeit. In diesem Kontext werden allfällige zentrale Lösungen wie Backup zu einem Cloud-Anbieter oder eine Ablage zu Hause nicht als dezentral angesehen, obwohl dort eine Mehrfachkopie und Verteilung möglich wäre. Eine Lösung gilt für diese Arbeit als dezentral sobald, ausser dem Besitzer, alle Parteien immer nur einen Teil oder Share des Keys, ihn jedoch nie in seiner Gesamtheit sehen. Die oben genannten Lösungen für Social Recovery oder TSS sind somit zu bevorzugen und werden besser bewertet.

### K6: «Application Use Cases»

Da die Arbeit grundsätzlich die Frage nach einer Key Recovery Lösung für den SSI Use Case beantworten soll, ist dieses Kriterium als Muss-Kriterium definiert. Dabei ist es in einem ersten Schritt auch gut möglich, dass die Applikation nicht speziell für SSI konzipiert wurde und später durch uns erweitert werden kann. Wenn die Applikation nicht speziell für den SSI Use Case erstellt wurde, muss sie zwingend das Kriterium OpenSource erfüllen oder zumindest eine Schnittstelle für SSI Applikationen bieten. Damit hätten wir die Möglichkeit, die Applikation so zu erweitern, dass SSI unterstützt wird oder eine Verknüpfung von mehreren Applikationen möglich ist.

## **K7: Recovery Use Cases**

Eine akzeptierbare Applikation sollte die nachfolgenden Use Cases abdecken. Als Mindestkriterium sollten die Punkte "Verlieren des Keys" und "Entität verlieren" realisiert sein.

### **K7a: Verlieren des Keys**

Der Besitzer des Keys verliert das Device, auf welchem der Key gespeichert ist. Sei dies ein Smartphone, eine portable Disk oder ein sonstiges Gerät. Unter diesen Punkt fällt zudem, dass der Key kompromittiert wird und in die Hände von Unbefugten gelangt ist.

Für einen Enduser kann es schwierig sein abzuschätzen ob sein Gerät gestohlen wurde oder "nur" verloren ging. Generell sollte hier zwischen Keys für Verschlüsselung und Keys für Signaturen unterschieden werden. Falls das Gerät auf jeden Fall verloren oder kaputt ging und ausgeschlossen werden kann, dass ein Dritter darauf zugreift, macht es in beiden Fällen Sinn den Key wiederherzustellen. Für Keys, welche man für Verschlüsselung einsetzt, hat man nur eine Option. Man kommt nicht darum herum diesen wiederherzustellen, selbst wenn das Gerät in die Hände eines Dritten gelangt ist. Wenn allerdings ein Key für Signaturen auf dem Gerät liegt sollte eine Revozierung durchgeführt werden. Diese ist allerdings klar vom «Application Use Case» abhängig und deshalb nicht im Scope dieser Arbeit. Wie im Kapitel «Mögliche Lösung» beschrieben gehen wir von einem verschlüsselten Wallet aus, welches wir sichern und wiederherstellen wollen. Somit ist der Key, welcher verteilt wird, für eine Wiederherstellung zwingend notwendig und kann nicht ersetzt werden.

### **K7b: Entität verlieren**

Eine Entität, welcher man einen Share des Keys anvertraut hat, ist nicht mehr vertrauenswürdig oder existiert nicht mehr (Person stirbt, Firma geht Konkurs usw.). Wie dem Besitzer, könnte auch der Entität das Device abhandengekommen sein. Somit sollte es eine Möglichkeit geben, verteilte Shares zu entziehen oder diese ungültig zu machen.

### **K7c: Besitzer existiert nicht mehr**

Hierbei spielt der rechtliche Aspekt eine Rolle. Was passiert mit dem Key, wenn der Besitzer nicht mehr existiert (eine Firma Konkurs geht, Besitzer stirbt usw.). Die Applikation soll diese Möglichkeit behandeln und eine Lösung dafür bieten.

## **K8: Kosten**

Als erster Punkt sollte die Applikation kostenfrei zur Verfügung stehen. Zweitens sollten durch das Recovery oder das Verteilen des Keys per se keine Kosten aufkommen. Eine allfällige Motivation für das sichere Aufbewahren von Shares, z.B. in Form einer Entlohnung der Teilnehmer, kann aber unter Umständen sehr sinnvoll sein. Insbesondere wenn es sich bei einem konkreten Teilnehmer um eine Organisation oder Firma handelt, ist nicht davon auszugehen, dass die Aufbewahrung der Shares und die Unterstützung beim Recovery-Fall gar keine Kosten generieren wird. Deshalb ist der zweite Punkt auch nicht von grosser Bedeutung, wird aber in die Gegenüberstellung der Applikationen einfließen.

## 5 Applikationen

### 5.1 Connect.me

Wiederherstellungsmethode: Mnemonic

Version: 1.2.0

Bei Connect.me handelt es sich um eine kostenfreie closed-source Mobile App für iOS und Android, welches Identitätsbeziehungen im SSI Umfeld abspeichern und verwalten soll. Also um ein "Digital Identity Wallet". Somit passt es perfekt in den Use Case SSI rein. Das Wallet wurde von der Firma Evernym erstellt, welche eine Vorreiterrolle im ganzen SSI Umfeld einnimmt.

Obwohl Evernym sowohl hinter dem Sovrin Netzwerk, der Sovrin Foundation und dem Hyperledger Indy Projekt steht, und letzteres in seinem Wiki neben Mnemonic auch von Social Recovery spricht, ist in diesem App nur das Recovery per Recovery Phrase, sprich Mnemonic, unterstützt. Dabei kann das ganze Wallet, als .zip verschlüsselt und anschliessend manuell gesichert oder automatisch in die Cloud hochgeladen werden. Verliert man sein Gerät und möchte das Wallet auf einem neuen Gerät installieren, holt man sich die .zip Datei, gibt die Recovery Phrase ein und schon hat man sein Wallet zurück. Der Prozess ist somit relativ einfach zu navigieren, verlässt sich aber komplett auf eine korrekte Aufbewahrung der Phrase. In den Einstellungen der App ist einfach ersichtlich, ob das automatische Cloud Backup eingeschaltet ist und falls ja, wann dieses das letzte Mal durchgeführt wurde. Ob das Backup alle aktuellen Änderungen enthält und ob in letzter Zeit ein manuelles Backup durchgeführt wurde ist nicht ersichtlich. Beim Backup wird dem User aber erklärt, dass er für das Recovery unbedingt die Phrase braucht, welche auch in den Einstellungen abgelesen werden kann. Eine Wiederherstellung per Recovery Phrase ist generell nicht dezentral, weshalb das Kriterium K5 nicht erfüllt ist. Weiter ist auch nicht geklärt was passiert, wenn ein User sterben würde. Allerdings scheint hier die offensichtliche Lösung, dass der User selbst verantwortlich ist, einem möglichen Erbe oder Anwalt Zugriff auf seine Recovery Phrase zu gewähren.

### 5.2 uPort

Wiederherstellungsmethode: Mnemonic

Version: 1.6.482

Bei der uPort App handelt es sich wie bei Connect.me um ein kostenloses "Digital Identity Wallet", welches nur als Mobile App zur Verfügung steht und OpenSource ist. uPort basiert auf Ethereum und deswegen arbeitet auch diese App mit der klassischen zwölf Worte langen Recovery Phrase.

Auch hier kann im App die Phrase angezeigt und bestätigt werden. Im Falle einer Wiederherstellung wird dann beim ersten Öffnen der App gefragt, ob man eine Wiederherstellung vornehmen will. Man gibt die 12 Worte ein, damit wird der Private Key berechnet und schon ist man zurück im Wallet. Anders als bei Connect.me gibt es hier keine Möglichkeit das Wallet offline zu sichern. Im App kann lediglich ein automatisches (verschlüsseltes) Backup des Wallets auf die Server von uPort aktiviert werden. Wie oft das geschieht und ob das Backup überhaupt funktioniert ist für den User nicht ersichtlich. Natürlich wird auch hier das Kriterium K5 nicht erfüllt, da es sich beim Recovery nicht um eine dezentrale Lösung handelt. Für den Fall, dass ein User sterben würde gilt auch hier dasselbe wie bei Connect.me.

### 5.3 Vault12

Wiederherstellungsmethode: Social Recovery

Version: 1.2.26

Vault12 steht für die gängigen Mobile-Plattformen iOS und Android und als Desktop-App für Windows und MacOS zur Verfügung. Vault12 hat keinen speziellen «Application Use Case», sondern kann alle möglichen Dateitypen mit Shamir Secret Sharing (Social Recovery) an vertraute Entitäten (Guardians) verteilen. Es ist dabei egal, ob es sich um Bilder, Seed Phrases, Kryptografische Schlüssel oder Passwörter handelt.



Abbildung 5: Vault12

Man kreiert zuerst einen eigenen “Vault” (Tresor) und wählt eines der folgenden Abonnements aus:

Free	Standard	Cryptopro
1 Asset (5MB)	5 Assets (50MB)	100 Assets (1GB)
Vaults laufen nach 30 Tagen ab und müssen neu hinzugefügt werden	Vaults laufen nicht ab	Vaults laufen nicht ab
Nur Mobile	Mobile / Desktop	Mobile / Desktop
	Custom Guardian Security Policy	Custom Guardian Security Policy

Tabelle 2: Vault12 - Abonnements

Hat man nun das gewünschte Abo gewählt, wird nach der Anzahl Guardians gefragt, die gebraucht werden, um den Zugriff wiederherzustellen: 3 von 5, 5 von 8, 7 von 10 oder bei den kostenpflichtigen Versionen kann das N von M selbst festgelegt werden. Anschliessend wählt man die entsprechende Anzahl Guardians aus dem Telefonbuch aus und sendet Einladungslinks per Mail oder SMS.

Nach der Bestätigung der Guardians können Fotos, Dateien oder Passwörter, genannt Assets, ausgewählt werden, die unter den Guardians verteilt werden. Diese Assets sind nach dem Verteilen gesperrt, können nur mit Bestätigung der N Guardians entsperrt und gelesen werden, und werden automatisch nach 2 Stunden wieder gesperrt.

Ein Ersetzen eines bestehenden Guardians ist jederzeit möglich und somit kann das Kriterium K7b vollumfänglich erfüllt werden.

Ein Recovery funktioniert wie das Entsperren eines Assets. Man benötigt N Guardians, welche per Telefonanruf oder SMS aufgefordert werden, einen Link für die Wiederherstellung zu schicken oder man scannt den QR Code, welcher auf der Applikation der Guardians erscheint. Es ist ausserdem auch möglich mehrere eigene Devices als Guardians anzugeben. Es muss jedoch beachtet werden, dass diese Devices nie zur selben Zeit am selben Ort sein sollten.

Die Applikation Vault12 ist leider nicht OpenSource und setzt ein funktionierendes Mobile Device voraus. Die Desktop Versionen können nur an die Mobile Devices gebunden werden und werden nicht als eigenes Device akzeptiert.

Es gibt eine ausführliche technische Dokumentation anhand eines Whitepapers (Max Skibinsky 2018) und diverse Videos, die zeigen, wie die App bzw. die Kryptographie in der App funktioniert.

### 5.4 Argent

Wiederherstellungsmethode: Social Recovery

Version: 1.0.1302

Argent Labs stellt mit Argent ein mobiles Wallet für die Kryptowährung Ethereum zur Verfügung, welche nicht OpenSource ist und auf Smart Contracts basiert. Für den Recovery-Fall oder das Überschreiten der täglichen Limite an Transaktionen wird die Bestätigung der sogenannten Guardians benötigt. Guardians können bei Argent im Gegensatz zu Vault12 nebst persönlichen Kontakten auch Hardware Wallets oder Argent selbst sein.

Der Wiederherstellungs-Prozess nur mit dem Argent Guardian ist mit einer Bestätigung per SMS und E-Mail möglich, was wiederum keinen dezentralen Mechanismus darstellt.



Abbildung 6: Argent

Für die Wiederherstellung mit zusätzlichen Guardians wird jeweils die Mehrzahl der Guardians benötigt:

Anzahl Guardians	Benötigte Anzahl
1	1
2	1
3	2
4	2
N	$\left\lceil \frac{n}{2} \right\rceil$

Tabelle 3: Argent - Anzahl Guardians

Bei Abhandenkommen des Gerätes kann ein Guardian das Wallet des Inhabers zudem für fünf Tage sperren oder entsperren, falls es fälschlicherweise gesperrt wurde. Sobald die fünf Tage abgelaufen sind, wird das Wallet automatisch entsperrt. Nach einer Sperre kann der Inhaber des Wallets dieses mit der Wiederherstellungs-Funktion auf einem neuen Gerät wiederherstellen.

Die Wiederherstellung dauert immer 36 Stunden und kann jederzeit unterbrochen werden, falls diese mit falscher Absicht angestoßen wurde. Für den Abbruch braucht es dann  $\left\lceil \frac{n+1}{2} \right\rceil$  Guardians.

Ein Transfer des Wallets auf ein neues Gerät ist jederzeit und ohne Zeitverzögerung möglich.

Die detaillierten Spezifikationen sind online zu finden. (argentlabs 2020)

## 5.5 Dark Crystal

Wiederherstellungsmethode: Social Recovery

Version: 8.1.0 (Patchbay)

Dark Crystal läuft auf dem Scuttlebutt Protokoll, einem dezentralisierten sozialen Netzwerk. Die Entwickler von Dark Crystal wollten mit ihrem Feature Kryptowährungen für den normalen User zugänglicher machen, indem sie das Key Recovery drastisch vereinfachen. Neben Kryptografischen Schlüssel können aber im Prinzip alle Arten von Strings, wie zum Beispiel Passwörter, so gesichert werden. Dark Crystal arbeitet mit Shamir Secret Sharing für das Splitten und zusammenfügen der Shares. Da das Scuttlebutt Netzwerk und dementsprechend auch das Dark Crystal Feature mit Peer-to-Peer Verbindungen arbeiten, werden die Shares direkt mit den gewünschten Freunden ausgetauscht, ohne dass sie vorher durch einen zentralen Server verarbeitet werden müssen.

Das Scuttlebutt Netzwerk kann zwar sowohl auf mobilen Geräten als auch auf dem Desktop verwendet werden.

Das Feature Dark Crystal steht aber nur in der "Patchbay"-Version von Scuttlebutt zur Verfügung, welche nur auf Windows, MacOS und Linux erhältlich ist.

Alle Funktionalitäten sind komplett kostenlos und der Source Code steht auf Github öffentlich zur Einsicht zur Verfügung.

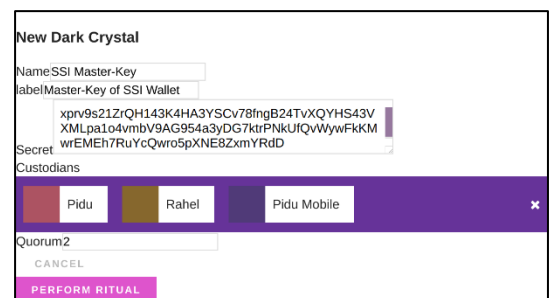


Abbildung 7: Dark Crystal

Soweit scheint Dark Crystal eine relativ passable Lösung zu sein, allerdings bestehen leider auch eindeutige Nachteile:

- Die Applikation ist instabil und nicht immer verständlich. Es treten immer wieder technische Fehler auf und die Applikation ist nicht intuitiv zu bedienen. Auch werden dem User die Beschränkungen und die Funktionalität der Applikation überhaupt nicht verständlich dargestellt. Es kann zum Beispiel ein Share eines Secrets an ein Mobile-Client verteilt werden, obwohl dieser damit gar nichts anfangen kann. Selbst als technikaffiner Nutzer war die Bedienung der Applikation sehr anstrengend.
- Das Scuttlebutt Protokoll verlässt sich seinerseits auf ein bestimmtes File, welches unter anderem den Private Key des Users enthält. Ohne dieses File verliert man seinen Scuttlebutt-Account und somit hat man wieder einen Single Point of Failure mit diesem File. Die Entwickler haben angekündigt, dass dies irgendwann auch per Dark Crystal verteilt werden soll.



- Das Entfernen bzw. Austauschen von Personen, an welche man einen Share gesendet hat, ist nicht möglich.

## 5.6 ZenGo

Wiederherstellungsmethode: Biometrics, Key Escrow

Version: 2.12.0



Abbildung 8: ZenGo

Die ZenGo App ist eine neuartige mobile App für Kryptowährungen. ZenGo behauptet von sich selbst, die erste Kryptoapp zu sein, die ohne Private Key auskommt (Keyless security). ZenGo verwendet threshold-basierte Signaturen, um Transaktionen zu signieren. Ein Teil des Schlüssels ist auf dem Mobile Device (Client Share) und der zweite Teil auf den Servern von ZenGo (Server Share) gespeichert. Dabei wird ein "2 von 2" ECDSA (Shlomovits, Introducing Multi-Party ECDSA library 2019), welcher OpenSource ist und auf GitHub (Networks 2020) auditiert werden kann, verwendet.

Es sind sehr ausführliche Dokumentationen (Be'ery 2019) zu Fragen wie "Was passiert, wenn das Smartphone verloren geht? Was ist, falls ZenGo Konkurs geht? Was passiert, wenn das Client oder Server Share verloren geht?" vorhanden.

Die eingerichtete Sicherung ist eine Bedingung für das Benutzen der Applikation. Für die Sicherung und die Wiederherstellung wird ein Verfahren mit dem Namen "ZoOm - 3D Face Authentication" von *facetec* (facetec 2020) verwendet. Dabei wird das Wallet inklusive dem Client Share verschlüsselt auf dem eigenen iCloud oder Google Drive Konto abgelegt. Der Verschlüsselungs-Schlüssel, welcher auf den ZenGo Server liegt, wird mit der "3D Face Authentication" geschützt.

Zusätzlich zum eigenen "Gesicht" kann auch ein zweites Gesicht hinterlegt werden.

Bei einer Wiederherstellung kann mit der "3D Face Authentication" der Verschlüsselungs-Schlüssel vom Server geholt und auf dem privaten iCloud oder Google Drive Konto das verschlüsselte Wallet wieder heruntergeladen werden.

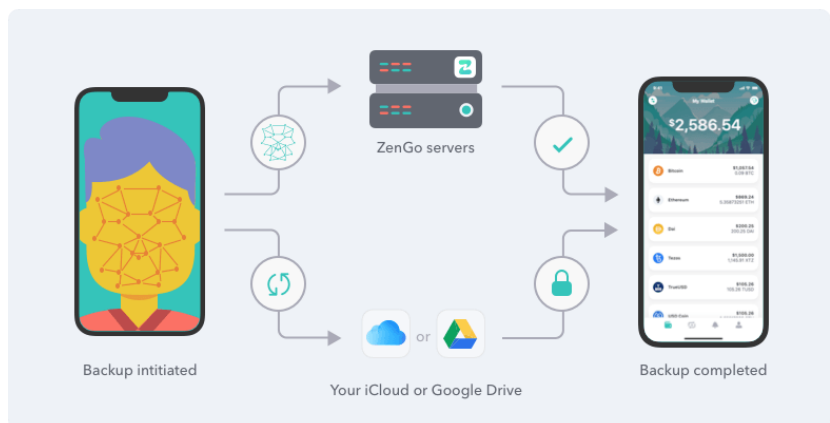


Abbildung 9: ZenGo - Backup

Somit schützt man sein Wallet einerseits mit dem Passwort des iCloud oder Google Kontos, wie auch mit seinen biometrischen Daten.

Die Server Shares sind weniger anfällig für einen Verlust und werden auf den ZenGo Servern gut geschützt. Doch was passiert, wenn das Unternehmen ZenGo Konkurs geht oder die Server kompromittiert werden?

Dazu sind die folgenden Vorkehrungen getroffen:

- Die Server Shares sind alle mit einem Master-Schlüssel verschlüsselt und dieser Schlüssel ist bei einem Escrow (EscrowTech 2020) hinterlegt.
- Die verschlüsselten Server Shares sind jeweils auf dem dazugehörigen Mobile Device abgelegt.



Abbildung 10: ZenGo - Recovery Mode

Eine Anwaltskanzlei prüft regelmässig, ob ZenGo funktionsfähig ist. Falls dies nicht der Fall ist, fordert die Kanzlei den Schlüssel beim Escrow an und publiziert diesen auf GitHub. Die Mobile App überwacht konstant den GitHub Account, entschlüsselt im Notfall die Server Shares und erstellt neue asymmetrische Schlüssel für das Kryptogeld. Somit ist man komplett unabhängig von ZenGo und kann das Kryptogeld in ein neues Wallet übertragen.

## 5.7 Bewertung

### 5.7.1 Punkteverteilung

Die, im Kapitel Kriterienkatalog definierten, Muss-Kriterien werden doppelt gewichtet, um deren Wichtigkeit in der Bewertung der Applikationen widerzuspiegeln. Die nicht notwendigen Kriterien, welche nur der Vollständigkeit halber erwähnt werden, werden mit einer 0 gewichtet.

Die Punkte sind wie folgt verteilt:

Punkte	Bedeutung	Beispiele
0	Gar nicht erfüllt	
1	Praktisch nicht erfüllt	K4: Nur ein Demo Use Case vorhanden K7b: Nur Konkurs des Anbieters ist behandelt K6: Nicht konkret für SSI, bietet aber eine Schnittstelle
2	Teilweise erfüllt	K2: Verfügbar nur auf Desktop K4: Dokumentation mangelhaft
3	Mehrheitlich erfüllt	K2: Verfügbar nur auf Mobile K8: Kostenfrei mit Limitierungen
4	Vollständig erfüllt	

Tabelle 4: Punkteverteilung

### 5.7.2 Bewertungsmatrix

Kriterium	G*	Connect.me	uPort	Vault12	Argent	Dark Crystal	ZenGo
K1: Benutzerfreundlichkeit	1	3	2	4	3	2	4
K2: Verfügbarkeit	1	3	3	3	3	2	3
K3: OpenSource	1	0	4	1	1	4	1
K4: Dokumentation	1	1	2	2	3	1	3
K5: Key Recovery	2	0	0	4	2	4	1
K6: «Application Use Case»	2	4	4	1	0	0	0
K7a: Verlieren des Keys	2	4	4	4	4	2	4
K7b: Entität verlieren	2	0	0	4	4	0	1
K7c: Besitzer existiert nicht mehr	0	0	0	0	0	0	0
K8: Kosten	1	4	4	3	4	4	4
<b>Gesamtpunktzahl</b>		<b>27</b>	<b>31</b>	<b>39</b>	<b>34</b>	<b>25</b>	<b>27</b>

Tabelle 5: Bewertungsmatrix

\* Gewichtung des Kriteriums

### 5.8 Fazit

Gemäss Punktzahl gibt es eine Applikation, die besonders ins Auge sticht. Vault12 setzt dank Social Recovery und dem Peer-to-Peer Mechanismus praktisch alle unseren Vorgaben um. Zudem ist sie extrem Benutzerfreundlich. Dies ist auch dem geschuldet, dass Vault12 nur als Tresor für Assets konzipiert ist und nicht einen konkreten «Application Use Case» unterstützt. Nebst dem, dass die Applikation nicht OpenSource ist, ist das auch schon das grösste Problem. Immerhin wird eine Schnittstelle für andere Apps zur Verfügung gestellt, dank welcher Vault12 fürs Aufbewahren von arbiträren Daten eingesetzt werden kann. (Vault12 2020)

Argent ist gut für Endanwender dokumentiert und setzt teilweise Social Recovery ein. Da die Anzahl Guardians nicht festgelegt ist und Argent selbst als Standardguardian hinterlegt ist, ist der dezentrale Ansatz leider nicht 100% garantiert und muss vom Endanwender selbst bestimmt werden. Dies ist bei nicht-technischen Endanwendern nicht die beste Voraussetzung. Weiter ist es auch unmöglich Argent für den gewünschten SSI Use Case einzusetzen, da sich die Funktionalität auf das Aufbewahren von Ethereum beschränkt.

uPort folgt als erste SSI- und OpenSource-Applikation dicht hinter den beiden oben genannten Applikationen. uPort könnte somit mit einem geeigneten dezentralen Wiederherstellungsprozess ergänzt werden und würde dann alle wichtigen Kriterien erfüllen.

## 6 “Showcase”

Die punktbeste Applikation der Bewertungsmatrix, Vault12, kann konkret dazu verwendet werden z.B. eine Seed Phrase abzulegen und später wiederherzustellen. So kann die Applikation theoretisch eingesetzt werden, um ein SSI Master-Key zu sichern.

Zu den Punkten im "Showcase" sind im Anhang Showcase die dazugehörigen Bilder abgelegt, welche in der Form (*Nummer*) referenziert werden.

### Aufsetzen des Vaults

Beim ersten Starten der Applikation kann man sich für eine der drei folgenden Optionen entscheiden:

- Eröffnen eines neuen Vaults (1)
- Beitreten bzw. Schützen eines bestehenden Vaults
- Recovery

Nachdem man sich für das Erstellen eines neuen Vaults entschieden hat, kann der Threshold gewählt werden. Wie im Kapitel Vault12 beschrieben ist man etwas eingeschränkt, wenn man für den Dienst nicht zahlen möchte. Für das Beispiel wird 3 von 5 gewählt. (2)

Der Besitzer kann nun entweder per Links oder per QR Codes seine Guardians hinzufügen. (3)

Sobald alle benötigten Guardians geantwortet und ihre Teilnahme bestätigt haben, ist das Aufsetzen abgeschlossen. (4)

### Sichern eines Assets

Nach dem Aufsetzen kann direkt damit begonnen werden die gewünschten Assets einzutragen. Im Beispiel wird eine “Note” in Form eines Seed Phrases mit dem Namen “Bitcoin.txt” gespeichert. (5)

Das neue Asset wird nun in der Liste der Assets aufgeführt und an die Guardians verteilt. (6)

### Ersetzen eines Guardians

Falls das Vertrauen in einen Guardian verloren gehen sollte, kann dieser ohne Probleme wieder per QR Scan oder per Link ersetzt werden. (7)

### Recovery

Falls man sein Smartphone verliert, muss man natürlich in der Lage sein den Vault und somit die verteilten Assets wiederherzustellen. Dazu wird nach der Neuinstallation des Apps die dritte Option “Restore a Vault” ausgewählt. (8)

Um die Bestätigung der Guardians zu erhalten, kann auch wieder mit QR Scans oder mit Links gearbeitet werden, welche die entsprechenden Guardians bestätigen müssen. (9)

Sobald genügend Guardians die Wiederherstellung bestätigt haben, ist der Vault wiederhergestellt. (10)

### Aufrufen eines Assets

Auch das Anschauen eines Assets wird durch Guardians reguliert. Wenn das bereits hinzugefügte Asset “Bitcoin.txt” angeschaut wird, muss es dafür entsperrt werden. Auch hier müssen wieder mindestens drei von fünf Guardians ihr OK geben. Nach erfolgreichem Entsperrern wird das Asset nach 2 Stunden wieder gesperrt. (11)

### Beschützen eines Vaults

Bisher wurde die Seite des Besitzers beleuchtet, nun soll die Seite des Guardians gezeigt werden. Generell kann jeder Guardian auch ein Besitzer und umgekehrt sein.

Der Guardian kann beim Inbetriebnehmen der App entweder die Option "Join an Existing Vault" auswählen oder er gelangt direkt mit einem Link, welcher er vom Besitzer erhalten hat, ins korrekte Menü. (12)

Als nächstes muss das "Guardian commitment level" ausgewählt werden. Damit wird bestimmt ob man vom Besitzer des Vaults eine Entschädigung erwartet. Für das Beispiel wurde "Pro-Bono" gewählt. (13)

Wenn der Besitzer ein Asset entsperren will, erhält ein Guardian eine Push-Benachrichtigung, welche im App bestätigt werden muss. (14)

Damit der Besitzer sich sicher sein kann, dass seine Assets weiterhin von allen Guardians geschützt werden, erhalten diese von Zeit zu Zeit eine Push-Benachrichtigung, welche bestätigt werden muss. (15)

## 7 Schlussfolgerungen

Das Kapitel Self-Sovereign Identity zeigt auf, wieso SSI das Identitätsmanagement der Zukunft ist oder zumindest sein könnte. Der dezentrale Ansatz verbessert sowohl die Sicherheit, als auch die Unabhängigkeit des Users. Allerdings ist es von immenser Bedeutung eine dementsprechend gute, sichere und trotzdem einfache Lösung für das Problem des Key Recovery zu finden, damit dem breiten Einsatz der SSI Technologie nichts mehr im Weg steht.

In der Gegenüberstellung hoben sich klar die dezentralen Ansätze "Social Key Recovery" und "Threshold Signatures" von den anderen Möglichkeiten ab. Auch wenn beim Einsatz dieser Methoden wieder neue Fragen und Probleme aufkommen, welche manchmal mehr und manchmal weniger schwierig zu lösen sind, sind sie doch dank ihrer Fehlertoleranz ("M von N"-Prinzip), ihrer Einfachheit und ihrer Sicherheit sehr geeignete Lösungen für das Key-Recovery-Problem von SSI.

Obwohl die Anforderungen relativ klar waren, stellte sich das Finden von passenden Applikationen als schwierig heraus. Das Konzept Social Recovery ist zwar alles andere als neu, wird jedoch kaum von Applikationen genutzt und wenn doch, dann nicht zwingend für einen passenden Use Case. Mit der Lösung namens "Vault12" kristallisierte sich trotzdem ein relativ geeigneter Kandidat heraus, mit welchem ein tatsächlicher Schlüssel auf gewünschte Entitäten verteilt und wiederhergestellt werden kann (siehe Kapitel "Showcase"). Aufgrund des nicht einsehbaren Quellcodes, der mittelmässigen Dokumentation und der entstehenden Kosten handelt es sich aber auch hier nicht um die perfekte Lösung für SSI.

Schlussendlich lässt sich also sagen, dass es bisher keine dezentrale Lösung für das Backup und Recovery von SSI Schlüsseln gibt. Bestehende SSI Applikationen setzen in der Regel auf das klassische "Mnemonic", mit welchem immer ein SPOF beim Enduser liegt.

## 8 Diskussion

Mit der Projektarbeit "Projekt 2" wird eine Grundlage und Vorbereitung geschaffen, um in einem nächsten Schritt eine konkrete Beschreibung und Umrahmung für eine Bachelorarbeit zu erstellen. In diesem Kapitel sollen die Ansätze und Ideen für eine Weiterführung erkundet werden.

Wie aus der Arbeit hervorgeht, existiert bisher keine Applikation, welche die Kriterien vollständig abdeckt und somit eine gute Lösung für das Key-Recovery-Problem von SSI darstellt. Es ergeben sich grundlegend zwei Möglichkeiten das Thema vertieft zu behandeln:

1. Anpassung einer bestehenden Applikation, um die gewünschten Kriterien abzudecken.
2. Entwickeln einer neuen Applikation, welche von Grund auf nach den ausgewählten Kriterien aufgebaut wird.

Auf beide Wege würde im Rahmen einer Bachelorarbeit ein Prototyp erstellt.

Die erste Möglichkeit, bei welcher eine bestehende Applikation angepasst würde, bietet den Vorteil, dass die vorhandene Funktionalität beibehalten wird und für den neuen Prototypen zur Verfügung steht. Beispielsweise bei uPort könnte so ein tatsächliches SSI App mit Key Recovery Funktionalitäten ergänzt werden. Dieser Vorteil ist aber gleichermassen auch ein riesiger Nachteil, da es enorm schwierig sein kann sich in unbekannten Code einzufinden und diesen nützlich zu ergänzen. Ausserdem muss auch die Programmiersprache dem Entwickler bereits bekannt sein, sonst müsste eine gewisse Sprache oder das Prinzip von Smart Contracts erst gelernt werden. Dies würde den Aufwand eine Applikation zu ergänzen massiv vergrössern.

Mit der zweiten Möglichkeit, einer eigenen, neuen Applikation, entfallen diese Punkte vollständig. Von Anfang an kann bestimmt werden, welche Funktionalitäten abgedeckt werden müssen. Diese können dann von Grund auf entwickelt werden, was auch dazu führt, dass der Code verständlich und für die Entwickler keine Einarbeitung in unbekannte Technologien und Sprachen von Nöten ist.

Um ein Social Recovery umzusetzen, müsste auf bestehende Libraries zurückgegriffen werden, da es schwierig ist, eine sichere, eigene Implementation zu schreiben. Die Shamir Secret Sharing Library von Daan Sprenkels wäre sicher eine gute Basis (Sprenkels 2019), da zum Beispiel auch HTC seine Zion App (Hank Chiu 2019) basierend auf dieser Library gebaut hat. Im Gegensatz zu anderen Libraries hat sich Daan Sprenkels genügend Gedanken über eine sichere Implementierung des Algorithmus gemacht und seine Library ist zum Beispiel auch resistent gegen Side Channel Angriffe. (Sprenkels 2019) Eine zusätzliche Herausforderung liegt beim Verteilen der Shares an vertrauenswürdige Entitäten. Dabei könnten in einem Prototyp die Teile per QR-Code verteilt werden, damit man nicht auf eine Peer-to-Peer Technologie oder eine Art öffentlichen Ledger angewiesen ist. Auch das Recovery könnte per QR-Codes umgesetzt werden.

Mit den gesammelten Erfahrungen und inspiriert von bestehenden Lösungen sollte es möglich sein eine zufriedenstellende Lösung - wenn auch vorerst nur in Form eines Prototyps - zu erstellen.

## 9 Abbildungsverzeichnis

Abbildung 1: Titelbild .....	1
Abbildung 2: Traditionelle Identität.....	5
Abbildung 3: Identity Provider.....	5
Abbildung 4: SSI .....	5
Abbildung 5: Vault12 .....	15
Abbildung 6: Argent .....	15
Abbildung 7: Dark Crystal.....	16
Abbildung 8: ZenGo.....	17
Abbildung 9: ZenGo - Backup.....	17
Abbildung 10: ZenGo - Recovery Mode.....	18

## 10 Tabellenverzeichnis

Tabelle 1: Gegenüberstellung SSI .....	6
Tabelle 2: Vault12 - Abonnements .....	15
Tabelle 3: Argent - Anzahl Guardians .....	16
Tabelle 4: Punkteverteilung.....	18
Tabelle 5: Bewertungsmatrix.....	19



## 11 Glossar

**DKMS**

Decentralized Key Management System. Ein Key-Management-Ansatz bei welchem es keine zentrale Verwaltung gibt.

---

**ECDSA**

Elliptic Curve Digital Signature Algorithm. Variante des Digital Signature Algorithms (DSA), welche Elliptische-Kurven-Kryptographie verwendet.

---

**Entropie**

Entropie ist ein Mass für den mittleren Informationsgehalt einer Nachricht. (Wikipedia 2020)

---

**Hardware Wallet**

Wallet bei dem der Private Key des Users in einem sicheren Hardware Gerät gespeichert wird.

---

**Patchbay**

Client Interface für das Scuttlebutt Protokoll mit experimentellen Features.

---

**Side Channel Angriff**

Angriff auf eine physische Implementierung eines kryptografischen Verfahrens. Nicht die Kryptografie wird angegriffen, sondern es wird versucht Korrelationen bei der Beobachtung zu finden. Zum Beispiel Laufzeit des Algorithmus, Energieverbrauch etc.

---

**Smart Contract**

Protokolle, um Verträge abzubilden, zu überprüfen und abzuwickeln.

---

**SPOF**

Single Point of Failure. Eine Stelle/Komponente dessen Ausfall das ganze Umsystem zum Erliegen bringt.

---

**TSS**

Threshold Signature Scheme

---

**Verifiable Credentials**

Credentials, welche digital signiert und verifizierbar sind.

---

**Zero-Knowledge-Proofs**

Ein Verfahren, bei welchem eine Partei der anderen beweist, dass sie ein bestimmtes Geheimnis kennt, ohne das Geheimnis selbst zu verraten.

---

## 12 Literaturverzeichnis


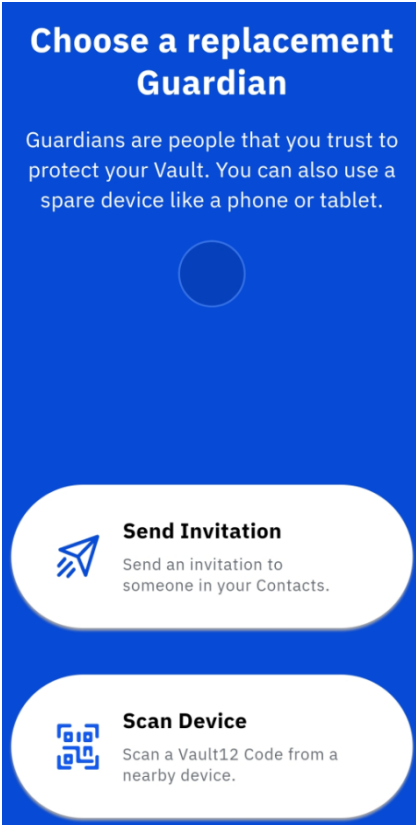
- argentlabs. «Argent Smart Wallet Specifications v1.6.» Mai 2020.  
<https://github.com/argentlabs/argent-contracts/blob/develop/specifications/specifications.pdf> (Zugriff am 22. Mai 2020).
- Be'ery, Tal. *How ZenGo guarantees access to customers' funds*. Juni 2019. <https://zengo.com/how-zengo-guarantees-access-to-customers-funds/> (Zugriff am Mai 2020).
- Bitcoin. *Bitcoin Improvement Proposals*. Juni 2020. <https://github.com/bitcoin/bips> (Zugriff am 07. Juni 2020).
- Christopher Allen, Mark Friedenbach. *A New Approach to Social Key Recovery*. 01. Februar 2019.  
<https://github.com/WebOfTrustInfo/rwot8-barcelona/blob/master/topics-and-advance-readings/social-key-recovery.md> (Zugriff am 14. April 2020).
- EscrowTech. 2020. <https://www.escrowtech.com/> (Zugriff am 07. Juni 2020).
- facetec. 2020. <https://www.facetec.com/> (Zugriff am 07. Juni 2020).
- Feldman, Paul. «A Practical Scheme for Non-interactive Verifiable Secret Sharing.» *IEEE Symposium on Foundations of Computer Science*, 1987: 427-437.
- Hank Chiu, Hankuan Yu, Justin Lin, Jon Tsai. «Social Key Recovery Design and Implementation.» Februar 2019. [https://github.com/WebOfTrustInfo/rwot8-barcelona/blob/master/topics-and-advance-readings/Social\\_Key\\_Recovery\\_design\\_implementation.md](https://github.com/WebOfTrustInfo/rwot8-barcelona/blob/master/topics-and-advance-readings/Social_Key_Recovery_design_implementation.md) (Zugriff am 01. Juni 2020).
- Max Skibinsky, Dr. Yevgeniy Dodis, Terence Spies, Wasim Ahmad. «Decentralized Storage of Crypto Assets.» Januar 2018. <https://s3-us-west-1.amazonaws.com/vault12/Vault12+Platform+White+Paper.pdf> (Zugriff am 17. Mai 2020).
- Mehmet Aydar, Salih Cemil Cetin, Serkan Ayyaz, Betül Aygün. *Private Key Encryption and Recovery in Blockchain*. Juli 2019. <https://arxiv.org/pdf/1907.04156.pdf> (Zugriff am 19. April 2020).
- Networks, KZen. *Rust implementation of  $\{t,n\}$ -threshold ECDSA (elliptic curve digital signature algorithm)*. Juni 2020. <https://github.com/KZen-networks/multi-party-ecdsa> (Zugriff am 07. Juni 2020).
- Pagliari, Emanuele. *BIP32, 39 and 44: differences between the most used seeds by wallets*. Juni 2019.  
<https://en.cryptonomist.ch/2019/06/01/bip32-39-44-differences-seeds-wallets/> (Zugriff am 14. April 2020).
- Pedersen, Torben Pryds. «Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing.» *11th Annual International Cryptology Conference*. Springer-Verlag, 1991. 129-140.
- Peg. *Security considerations for Shamir's secret sharing*. 31. Januar 2019.  
[https://github.com/WebOfTrustInfo/rwot8-barcelona/blob/master/topics-and-advance-readings/security\\_shamirs.md](https://github.com/WebOfTrustInfo/rwot8-barcelona/blob/master/topics-and-advance-readings/security_shamirs.md) (Zugriff am 14. April 2020).
- Ruff, Timothy. *The Three Models of Digital Identity Relationships*. April 2018.  
<https://medium.com/evernym/the-three-models-of-digital-identity-relationships-ca0727cb5186> (Zugriff am 13. April 2020).
- Schoenmakers, Berry. «A simple publicly verifiable secret sharing scheme and its application to electronic voting.» *Annual International Cryptology Conference*. Springer-Verlag, 1999. 148-164.
- Sean Gilligan, Gregory Jones, Adin Schmahmann, Andrew Hughes, Christopher Allen. *RWOT8: Evaluating social schemes for recovering control of an identifier*. August 2019.  
<https://github.com/WebOfTrustInfo/rwot8-barcelona/blob/master/final-documents/evaluating-social-recovery.pdf> (Zugriff am 14. April 2020).
- Sean Gilligan, Peg, Adin Schmahmann, Andrew Hughes, Christopher Allen. *RWOT8: Evaluating social schemes for recovering control of an identifier*. 02. August 2019.  
<https://github.com/WebOfTrustInfo/rwot8-barcelona/blob/master/final-documents/evaluating-social-recovery.md> (Zugriff am 11. Juni 2020).
- Shamir, Adi. «How to Share a Secret.» *Communications of the ACM*, 1979: 612-613.
- Shlomovits, Omer. *Introducing Multi-Party ECDSA library*. November 2019.  
<https://zengo.com/introducing-multi-party-ecdsa-library/> (Zugriff am 18. Mai 2020).
- . *Threshold Signatures Explained*. kein Datum. <https://academy.binance.com/security/threshold-signatures-explained> (Zugriff am 14. April 2020).
- Sprenkels, Daan. *Library for the Shamir secret sharing scheme*. September 2019.  
<https://github.com/dsprenkels/sss> (Zugriff am 06. Juni 2020).

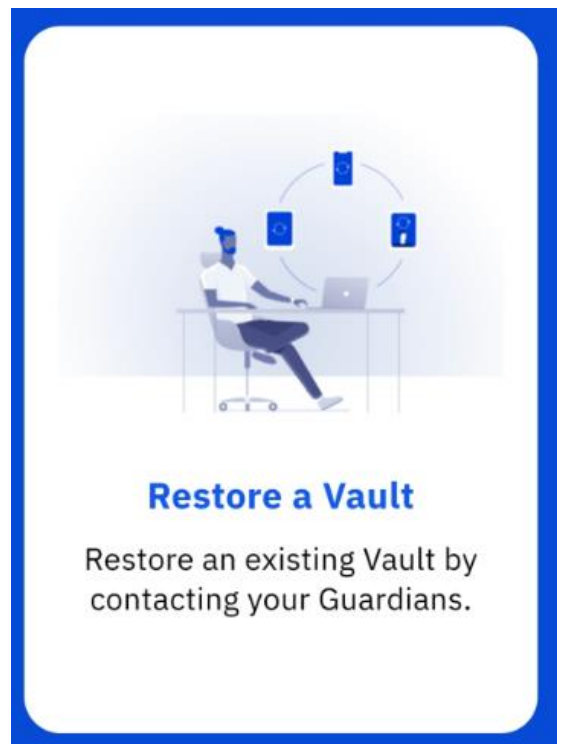
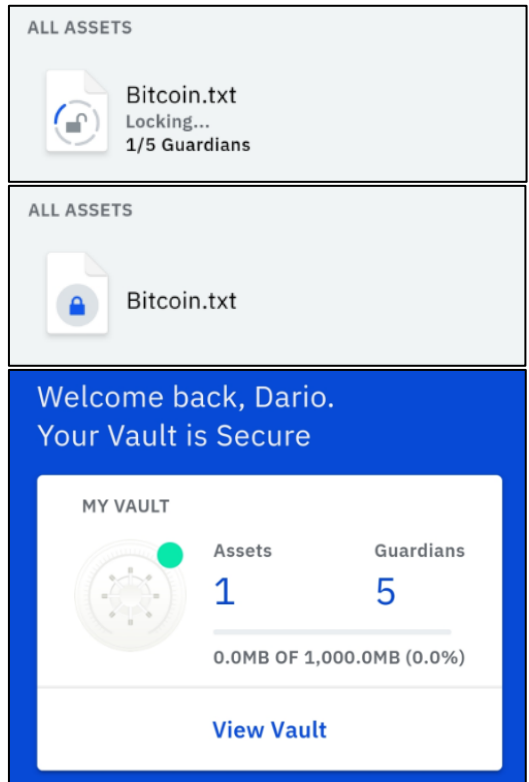
Vault12. *Update: April, 2020 Vault12 Latest Release Now Available*. 21. April 2020.  
<https://medium.com/vault12/update-april-2020-vault12-latest-release-now-available-df2e57d65278> (Zugriff am 11. Juni 2020).  
Wikipedia. *Entropie (Informationstheorie)*. 2020.  
[https://de.wikipedia.org/wiki/Entropie\\_\(Informationstheorie\)](https://de.wikipedia.org/wiki/Entropie_(Informationstheorie)) (Zugriff am 11. Juni 2020).  
ZenGo. *Security in-depth*. 2020. <https://zengo.com/security-in-depth/> (Zugriff am 07. Juni 2020).





## 13 Anhang








### 13.1 Showcase

1.		2.	
3.		4.	

5.		6.
7.		8.



9.	<div data-bbox="336 253 751 1072"> <h3>You still require confirmations...</h3> <p>More Guardians still need to respond before your Vault can be restored</p> <div> <div>BS</div> <div>LA</div> <div></div> </div> <div>  <p><b>Call a Guardian</b> Choose a Guardian from your Contacts to call.</p> </div> <div>  <p><b>Scan Device</b> Scan a Vault12 Code from a nearby device.</p> </div> </div>	10.	<div data-bbox="920 439 1458 887"> <h3>Restoring your Vault...</h3> <p>3 / 3 Guardians have responded.</p> <div> <div> <p>MY VAULT</p>  </div> <div> <p>Assets</p> <p>1</p> </div> <div> <p>Guardians</p> <p>5</p> </div> </div> <p>0.0MB OF 1,000.0MB (0.0%)</p> <p><a href="#">View Vault</a></p> </div>
11.	<div data-bbox="300 1149 791 1581"> <h2>Bitcoin.txt</h2> <p>This asset is locked. To unlock this asset, please contact <b>3 Guardians</b></p> <div>  </div> <p>Your asset is unlocked. It will automatically lock after 2 hours</p> </div>	12.	<div data-bbox="916 1205 1466 1525"> <h2>Dario Furigo invited you to be their Guardian!</h2> <p>As a Guardian, you will protect Dario Furigo's digital assets.</p> </div>

13.	<div data-bbox="300 257 1453 900"> <div>  <h3>Basic</h3> <p>You're just helping a friend with their Vault.</p> <hr/> <p>YOUR COMPENSATION:</p> <h2>Pro-Bono</h2> <p>Choose</p> </div> <div>  <h3>Advanced</h3> <p>You're Guarding a few Vaults. Part-time commitment.</p> <hr/> <p>YOUR COMPENSATION:</p> <div>  <h2>0.5</h2> <p>/yr*</p> <p>(\$99.89 /yr*)</p> <p>*Taxable Income ?</p> </div> <p>Choose</p> </div> <div>  <h3>Professional</h3> <p>You are guarding multiple Vaults with 24hr service.</p> <hr/> <p>YOUR COMPENSATION:</p> <div>  <h2>1</h2> <p>/yr*</p> <p>(\$199.78 /yr*)</p> <p>*Taxable Income ?</p> </div> <p>Choose</p> </div> </div>
14.	<div data-bbox="288 1010 796 1346"> <div>  <span>Vault12 • now</span> </div> <p>A Vault owner is requesting an unlock of an asset you are guarding</p> <div> <h4>Unlock Request</h4> <p>Dario Furigo is requesting an unlock of an asset you are guarding</p> <p><a href="#">View</a>   <a href="#">Dismiss</a></p> </div> </div>
15.	<div data-bbox="919 1113 1460 1276"> <div>  <span>Vault12 • 2h</span> </div> <p><b>Vault12</b></p> <p>Are you still protecting Dario Furigo's Vault? Tap here to confirm.</p> </div>

## 14 Selbständigkeitserklärung

Ich bestätige, dass wir die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt haben. Sämtliche Textstellen, die nicht von uns stammen, sind als Zitate gekennzeichnet und mit dem genauen Hinweis auf ihre Herkunft versehen.

Ort, Datum:

Unterschrift: