DKMS

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:	TODO

Management Summary

Dies ist eine Management Summary...

Inhaltsverzeichnis

1 Einleitung

Seit es Passwörter gibt, gibt es die Frage “Was, wenn ich mein Passwort verliere?”.
Deswegen sind wir alle vertraut mit den konventionellen Prozessen des Passwort-Recovery: Man wählt die Passwort-Vergessen Funktion aus, gibt seine E-Mail ein und erhält entweder einen Link für den PW-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, geht dies oftmals auf Kosten der User Experience.

Eine sichere Alternative zu Passwörtern bieten 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 Crypto-Währungen wie Bitcoin. Verliert man dort den Key zu seinem Wallet, sind jegliche Coins nicht mehr zugänglich.
Ein weiterer Use Case, welcher als Ausganslage dieser Arbeit diente ist “Self-Sovereign Identity (SSI)”. Verliert man dort sein Wallet und hat keinen Weg es zu restoren, sind alle Identitätsbeziehungen etc. verloren. (siehe Kapitel TODO)
Stellt man sich hier nun also dieselbe Frage “Was, wenn ich meinen Key verliere?”, 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 einfach wiedergeben kann.
Weiter stellt sich die Frage, was bei einem gestohlenen Private Key passiert. Habe ich eine Möglichkeit diesen zu revozieren oder auszutauschen?

Zusätzlich spielen andere Faktoren eine Rolle: Will ich einen verlorenen Key, welche für Signaturen verwendet wurde, überhaupt wiederherstellen, da ich ja nicht weiss, ob ihn jemand anderes entwendet hat und nun auch verwenden kann? Zum Entschlüsseln der Daten, kommt man jedoch nicht um eine Wiederherstellung herum.
Die Lösungen für diese Probleme sind vielseitig. Der am meisten verbreitete Ansatz zur Wiederherstellung des Private Keys ist eine Seed Phrase (auch Mnemonic genannt - siehe Kapitel TODO), 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 wird. (siehe Kapitel TODO)
Auch weit verbreitet ist ein zentrales Backup bei einem Provider. Dies ist jedoch nicht in unserem Sinn, da wir den Auftrag haben eine dezentrale Lösung zu suchen. Dabei ist das Stichwort DKMS (Glossar) von zentraler Bedeutung.

Diese und andere Methoden, die darauf basierten Anwendungen und das Problem als Ganzes wollen wir in dieser Arbeit von verschiedenen Seiten beleuchten.
Das Ziel wäre eine Lösung zu finden, welche für SSI nutzbar und dezentral gehalten ist.

2 Self-Sovereign Identity

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

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.

Vorteile: Einfachheit, weitverbreitet, keine Abhängigkeit von Dritten, einmalige Credentials wenn Username und Passwort nicht wiederverwendet werden.

Nachteile: Schlechte User Experience (hunderte Credentials, vergessene Passwörter...), schlechte Security (wiederverwendete Passwörter, Breaches haben gravierende Auswirkungen, Phishing...)

Third Party IDP (IDentity Provider)
In diesem Modell wird sich auf eine dritte Organisation verlassen, welche das Identitätsmanagement übernimmt und die Identität des Users an den entsprechenden Service "föderiert".
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 "social logins" wie "Facebook Login".

Vorteile: Einziges Credential, welches sicher gehalten und gepflegt werden muss. -> Mehr Sicherheit und bessere Experience.

Nachteile: Abhängigkeit von Dritten, IDP kann enorm viele Daten sammeln, Phishing möglich

Self-Sovereign / Peer-to-Peer
Mit dem SSI Modell gibt es keine Dritten im Identifizierungsprozess mehr.
Jeder Teilnehmer besitzt ein digitales Wallet, in welchem "Verifiable Credentials" (digital signierte verifizierbare 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.
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.

Vorteile: Stärkere Authentication (anstelle von Passwörtern werden kryptografische Keys verwendet), verbesserte User Experience, kein Phishing (gegenseitige Authentication), privater Kommunikationskanal (End-to-End Verschlüsselung), verbesserte Privatsphäre (es können dank Zero Knowledge Proofs (TODO GLOSSARY) nach Wunsch nur Teile eines Credentials gezeigt werden), Anonymität (zwei Organisationen können untereinander ihre Kunden nicht "linken").

Nachteile: Der Wechsel auf SSI generiert Kosten und muss erstmal weitverbreitet anerkannt werden. Ausserdem ist die Frage des Key Management noch nicht konkret gelöst. (siehe Kapitel 2.2 TODO)

https://medium.com/evernym/the-three-models-of-digital-identity-relationships-ca0727cb5186

2.2 Die ungelöste DKMS Frage

Wenn man bei einer traditionellen Webseite sein Passwort vergisst, gibt es immer noch die Möglichkeit von der "Passwort vergessen"-Funktion der Webseite Gebrauch zu machen.
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 gebackuped?
- 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

Um mehrere Keys bzw. ein Wallet zu schützen, bedienen wir uns dem "Passwortmanager Prinzip". Das heisst, wir verschlüsseln das Wallet als Ganzes und backupen es mit einer noch zu definierenden Technologie. So können wir das verschlüsselte Wallet und den dazugehörigen Master Key wiederherstellen.

3 Key Recovery Technologien

3.1 Mnemonic / Seed Phrase

Ein Mnemonic ist eine bestimmte Anzahl natürlicher Wörter, mit welchem ein Private Key deterministisch abgeleitet werden kann. Es gibt eine kleine Anzahl Standards (Bitcoin Improvement Proposals BIP): https://github.com/bitcoin/bips, welche beschreibt 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 secure.
https://en.cryptonomist.ch/2019/06/01/bip32-39-44-differences-seeds-wallets/

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 natürlich 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.
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, "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 und somit den Schnittpunkt mit der Y-Achse zu erhalten und das Secret zu restoren.
https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing
https://cs.jhu.edu/~sdoshi/crypto/papers/shamirturing.pdf

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

Gedanken
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 alleine 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.

- Verifizierbare Shares

Wenn bei einem Restore-Versuch der generierte Key nicht stimmt, gibt es für den Owner 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 Owner (“Dealer”) auch wirklich gültige Shares herausgegeben hat.
Zu den gängigsten Schemen zählen: Feldmann’s Scheme, Pederson Scheme und Schoenmaker’s Scheme.
(Feldman, Paul (1987) "A practical scheme for non-interactive Verifiable Secret Sharing" Proceedings of the 28th Annual Symposium on Foundations of Computer Science / Schoenmakers, Berry (1999) "A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting" Advances in Cryptology-CRYPTO'99, volume 1666 of Lecture Notes in Computer Science, pages 148-164, Berlin, 1999. Springer-Verlag.)

- Verlorener Trust / Entfernen einer vertrauten Entität

Verliert man das Vertrauen in eine oder mehrere Teilnehmer, muss es möglich sein, diesen aus dem Prozedere auszuschliessen. Beim Schema von Shamir können theoretisch unendlich viele Sets von Shares für das gleiche Secret erstellt werden. So könnte z.B. beim Vertrauensverlust ein neues Set erstellt und dieses nur noch an die neu vertrauten Parteien verteilt werden.

- Motivation der Teilnehmer

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

- Single Point of Compromise

Das Gerät, welches den Key aus den Shares zusammenstellt, stellt ein Security Risiko dar. Es müssen sich um Themen gekümmert werden wie: Wo wird die Prozedur durchgeführt? Bleibt danach etwas im Memory? Wo wird der resultierende Key gespeichert?

- Anonymität

Anonymität innerhalb der Teilnehmer kann gewünscht sein, da es so für diese massiv schwieriger wird einen Collusion-Angriff zu starten. Allerdings zieht das dann auch mit sich, dass es eventuell schwieriger wird z.B. bei einem Todesfall des Owners den Key wiederherzustellen.

- “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 die Teilnehmer regelmässig überprüft werden, ob der Share noch vorhanden ist.

- Rechtlicher Aspekt

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

- Welche Daten werden geshared?

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 sharen, um diese wiederherzustellen.

https://nbviewer.jupyter.org/github/WebOfTrustInfo/rwot8-barcelona/blob/master/final-documents/evaluating-social-recovery.pdf
https://github.com/WebOfTrustInfo/rwot8-barcelona/blob/master/topics-and-advance-readings/social-key-recovery.md
https://github.com/WebOfTrustInfo/rwot8-barcelona/blob/master/topics-and-advance-readings/security_shamirs.md

3.3 Paralysis Proof

Wollen sich N Geschäftspartner ein gemeinsames Wallet mit Cryptowährung teilen und Transaktionen sollen nur bei Zustimmung aller Geschäftspartner möglich sein, liegt es nahe, auf ein “N von N”-Multisig-Schemata [glossar: Mehrere Benutzer halten einen Teilschlüssel, der nur zusammen gültig ist] zu setzen.
Nun muss man sich aber auch die Fragen stellen “Was passiert, wenn jemand seinen Teilschlüssel verliert?”.
Dies will der “Paralysis Proof” Ansatz lösen, indem eine SGX-[glossar]basierte Applikation so programmiert wird, dass ein eindeutiger Beweis vorhanden sein muss, wenn ein Teilschlüssel verloren gegangen oder nicht mehr genutzt wird.
Dabei werden alle Keys bei den SGX-Komponenten hinterlegt. Weiter wird sie so programmiert, dass sie als TTP (TODO Glossar) agiert und im Falle eines korrekten Beweises die Signatur mit dem letzten, fehlenden Key übernimmt.
Der Beweis besteht darin, dass sich die Person mit dem Teilschlüssel, welche nicht mehr vorhanden ist, in einer gewissen Zeit nicht meldet. Dabei wird die eine Transaktion von N-1 Geschäftspartnern über die SGX-Applikation auf einer Blockchain festgehalten. Ist die Transaktion nicht im Sinne der letzten Person, kann die fehlende Person einfach ihr Schlüssel benutzen, um ein Veto einzulegen und es passiert nichts. Falls sich die fehlende Person nicht meldet, z.B. im Falle eines verlorenen Schlüssels, wird die SGX-Applikation die Transaktion signieren und somit freigeben.
Für uns könnte der Paralysis Proof wie folgt eingesetzt werden:
Es wird zu Beginn durch den Owner ein Public Key festgelegt, welcher im Recovery Fall dazu berechtigt ist öffentlich für den Owner einen neuen Key anzkündigen. Ab diesem Zeitpunkt wäre der alte Key dann ungültig.
Die Signatur dieser Ankündigung soll dabei durch den Paralysis Proof geschützt sein. Es könnten 3 Freunde des Owners über die dazugehörigen Private Keys verfügen. Wäre ein Freund nicht erreichbar oder hätte seinen Teil verloren, würde die SGX-Applikation die Signatur übernehmen.
Die Dauer, mit welcher auf den eventuell fehlenden Freund gewartet wird, kann hier aber zum Problem werden. Bei einer Kompromittierung des Geräts könnte in dieser Zeit gegebenenfalls schon viel Unheil angerichtet worden sein.

https://hackingdistributed.com/2018/01/18/paralysis-proofs/

3.4 Threshold Signatures / MultiSig

Threshold Signaturen finden ihren Einsatz primär für Transaktionen auf der Blockchain, welche aber von einer bestimmten Menge von Teilnehmer bestätigt werden muss.
Generell berechnen hier die Teilnehmer zusammen einen Public Key und halten jeweils einen eigenen Secret Share des Private Key. Die Signatur einer Transaktion wird nun von "M von N” Teilnehmern signiert, bevor sie gültig ist. So können (N-M) private Teile verloren gehen und die Transaktionen können trotzdem noch durchgeführt werden.
Wenn die Blockchain die TSS (Threshold Signaure Scheme TODO Glossary) als Teil ihrer Funktionalität unterstützt spricht man von “Multisig”.
Für den Use Case dieser Arbeit sind TSS 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.
https://www.binance.vision/security/threshold-signatures-explained
https://nbviewer.jupyter.org/github/WebOfTrustInfo/rwot8-barcelona/blob/master/final-documents/evaluating-social-recovery.pdf

3.5 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 z.B. 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 weiters Sicherheitsrisiko, da es sich ggf. um einen Dritten handelt, welcher so Zugriff auf private Schlüssel bekommt.

3.6 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.

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, welche die Sicherheit noch weiter erhöht.
Das grosse Problem bei den Biometrics ist, dass sich die Revocation 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 “zerstörter” Finger das Recovery nicht unmöglich machen.

https://arxiv.org/pdf/1907.04156.pdf

4 Kriterienkatalog

Die Applikation, welche wir suchen, sollte nachstehende Kritieren erfüllen bzw. diese zumindest auf der Roadmap haben.

4.1 Benutzerfreundlichkeit

Die wenigsten bekannten Lösungen für Key Recovery sind einfach zu handhaben. Deshalb ist es für uns ein wichtiges Kriterium, dass die Lösung auch durch normale Alltagsanwender und nicht nur durch Spezialisten bedienbar ist. Wenn wir uns im speziellen SSI als Use Case anschauen, gehen wir davon aus, dass in absehbarer Zukunft SSI für alle Menschen und nicht nur für technisch versierte Anwender eine Rolle spielen kann und auch soll.
Wir benötigen dazu:

- 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.

4.2 Verfügbarkeit für alle gängigen Plattformen

Die Lösung sollte für alle gängigen Plattformen erhältlich sein. Dabei sprechen wir von mobilen Betriebssystemen wie iOS und Android als auch von nicht-mobilen Betriebssystemen wie Linux, MacOS und Windows.

4.3 OpenSource

Der Code der Lösung 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 die Applikation anpassen. Somit setzen wir voraus, dass eine Lizenz verwendet wird, bei welcher wir die Applikation uneingeschränkt anpassen können und auch wieder als OpenSource Software veröffentlichen dürfen.

4.4 Dokumentation

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

4.5 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. Wir sprechen hier von einer dezentralen Lösung sobald, ausser dem Owner, alle Parteien immer nur einen Teil oder Share des Keys, jedoch diesen nie in seiner Gesamtheit sehen.

4.6 Application Use Cases

Da die Arbeit grundsätzlich die Frage nach einer Key Recovery Lösung für den SSI Use Case beantworten soll, sollte die Anwendung natürlich einen solchen unterstützen.
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 Verbindung von mehreren Applikationen zu realisieren.

4.7 Recovery Use Cases

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

4.7.1 Verlieren des Keys

Der Owner 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.
Die Unterscheidung dieser beiden Fälle (Verlust / Diebstahl) ist meist nicht einfach, kann aber zum Beispiel bei Signaturen von grossem Wert sein, bei welchen eine Ersetzung des Keys anstrebenswert wäre.
Wie im Kapitel mögliche Lösung (TODO) beschrieben gehen wir von einem verschlüsselten Wallet aus, welches wir backupen und wiederherstellen wollen. Somit ist der Key, welcher verteilt wird, für eine Wiederherstellung zwingend notwendig und kann nicht ersetzt werden.
Die Kompromittierung eines Keys hingegen setzt eine Revocation voraus, welche klar vom Application Use Case abhängig ist. Deshalb ist die Kompromittierung und somit Revocation nicht im Scope unserer Arbeit und wird nicht explizit verlangt.

4.7.2 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.). Der Punkt, welcher auch hier nicht zu vergessen ist, ist dass der Entität auch das Device abhandengekommen sein kann.
Somit sollte es eine Möglichkeit geben verteilte Shares zu entziehen oder diese ungültig zu machen.

4.7.3 Owner existiert nicht mehr

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

4.8 Kosten

Als erster Punkt sollte die Applikation kostenfrei zur Verfügung stehen. Zweitens sollte 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 Entlöhnung der Teilnehmer, erachten wir aber unter Umständen als sinnvoll. 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 Vergleich

5.2 Fazit

Was fehlt in den Applikationen und was können wir gebrauchen?

6 “Showcase”

7 Schlussfolgerungen/Fazit

Was sind mögliche Bachelorthesis Themen, wo können wir ansetzen/erweitern?

8 Abbildungsverzeichnis

9 Tabellenverzeichnis

10 Glossar

11 Literaturverzeichnis

Literatureintrag
Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr 7

Literatureintrag
Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr 9

Literatureintrag
Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr 11

12Anhang

Et ut aut isti repuditis qui ium nonsecturia quis incientiae laborem elliquis et quatur, sitiur aut od moluptatur aut ea conseque peri sim erro essequisit remporia dem et landi dest, cone poris quunt volecab ipidero quatur ad quibusamus.

13Selbständigkeitserklärung

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

Ort, Datum:

Unterschrift: