Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Entwurf vom
1. Oktober 2024

Bachelorarbeit

Privatsphärewahrendes Anreiz- und Betrugserkennungssystem im Datentreuhandmodell

vorgelegt von

Knut Hoffmeister
Matrikelnummer 7509085
Studiengang Software System Entwicklung

MIN-Fakultät
Fachbereich Informatik

eingereicht am 1. Oktober 2024

Betreuer: Kevin Röbert

Inhaltsverzeichnis

1 Einleitung **5**

2 Hintergrund **6**
2.1 Herkömmliche Datenkommunikation . . . 6
2.2 Datentreuhänder . . . 6
2.2.1 Definition . . . 7
2.2.2 Verschieden Modelle . . . 7
2.3 Anwendungsfälle . . . 8
2.4 Bestehende Anreize . . . 9

3 Anforderungen und verwandte Arbeiten **10**
3.1 Anforderungen . . . 10
3.1.1 Funktionale Anforderungen . . . 10
3.1.2 Nicht funktionale Anforderungen . . . 10
3.2 GNU Taler . . . 10
3.3 Bitcoin . . . 11
3.4 Ethereum . . . 12
3.5 Privacy Pass . . . 13
3.5.1 Umgehen vom CAPTCHAs . . . 13
3.5.2 Funktionsweise . . . 13
3.6 Privacy-Preserving Reputation Management . . . 15
3.7 Blinde Signaturen . . . 16
3.8 Partiell Blinde Signaturen . . . 17
3.8.1 Funktionweise . . . 17
3.9 Elliptische Kurven Kryptographie . . . 18
3.9.1 Trapdoor functions . . . 18
3.9.2 Funktionsweise . . . 19
3.9.3 Anwendung in der Kryptographie . . . 21
3.9.4 Vor und Nachteile . . . 21
3.10 AES - Advanced Encryption Standard . . . 21
3.11 Abschluss . . . 22

4 Entwurf der Systeme **24**
4.1 Ziel . . . 24
4.2 Geld als Anreiz . . . 24
4.3 Annahmen über das Datentreuhändermodell . . . 25
4.4 Generelles . . . 26
4.5 Verschlüsselung . . . 27
4.6 Coin Generierungsphase . . . 28
4.6.1 Ablauf . . . 28
4.6.2 Finanzierung des Exchange . . . 29
4.7 Bezahlvorgang . . . 29
4.7.1 Ablauf . . . 30
4.7.2 Verwendung von Proxys . . . 33
4.7.3 Bezahlungszeitraum . . . 33

4.7.4 Mögliche Streitfälle . . . 34
4.8 Reputationsvergabe . . . 36
4.8.1 Ablauf . . . 36

5 Implementation **37**
5.1 TRESOR Projekt . . . 37

6 Auswertung **38**
6.1 Bewertung der Sicherheit . . . 38
6.1.1 Angreifermodelle der Coin Generierungsphase . . . 39
6.1.2 Angreifermodelle der Bezahlphase . . . 41
6.2 Setup / Implementation . . . 41
6.3 Analyse des Rechenaufwands . . . 41
6.3.1 Metrics . . . 41

7 Bewertung **42**
7.1 Erfüllen der Anforderungen . . . 42
7.2 Wie kann ein privatsphäreschützender Anreiz zur Benutzung eines Datentreuhändermodells geschaffen werden? . . . 42
7.3 Wie kann dieser Anreiz gegen Missbrauch geschützt werden? . . . 42
7.4 (Discussion) . . . 42

8 Begrenzungen **43**

9 Zusammenfassung **44**

Abstract

1 Einleitung

In der heutigen Zeit wird der fachgerechte Umgang mit Daten jeglicher Form, zunehmend wichtiger. Viele der großen Player wie Facebook oder Google, machen ihr Hauptgeschäft mit dem Verwenden von Nutzerdaten zu gewerblichen Zwecken, wie bspw. targeted Ads [Fac24] [Goo24]. Und obwohl das Misstrauen eines Nutzers, solch eines großes Unternehmens gegenüber berechtigt ist, benötigen diese die gesammelten Daten auch, um neue Technologien zu entwickeln. Jedoch hat der Nutzer, der diese Daten generiert, meist keine Einsicht darüber, wer seine Daten verwendet und wofür diese zum Einsatz kommen. Ein potenzieller Lösungsansatz für dieses Problem, ist die Verwendung von Datentreuhändersystemen. In einem Datentreuhändersystem, kann ein Endnutzer der Daten generiert zum einen seine Daten verschlüsselt bei einem Treuhänder lagert. Zum anderen ist der Treuhänder ein Mediator zwischen dem Endnutzer und einem Unternehmen, welches an den Daten interessiert ist. Möchte nun ein Unternehmen die Daten für ihre Zwecke verwendet, so kann dieses bei dem Treuhänder Daten anfordern. Daraufhin kontaktiert der Treuhänder den Nutzer, von dem die Daten stammen und fragt diesen, nach seiner Zustimmung. So kann der Nutzer genau einsehen, wer seine Daten verwenden möchte und kann unerwünschte Benutzung unterbinden. Das Ziel liegt hierbei vor allem darauf den Nutzer und seine Privatsphäre so gut wie möglich zu schützen.

In dieser Arbeit sollen, um die Motivation zur Benutzung eines Datentreuhänder zu erhöhen, zwei Forschungsfragen beantwortet werden: Wie kann ein privatsphäreschützender Anreiz zur Benutzung eines Datentreuhändermodells geschaffen werden? Und: Wie kann dieser Anreiz gegen Missbrauch geschützt werden? Aktuell ist der einzige Anreiz zur Benutzung eines Datentreuhändermodells für den Nutzer, der Schutz der Privatsphäre durch die Kommunikation über den Treuhänder. Um einen weiteren Anreiz zu bieten, soll hier ein Ansatz vorgestellt werden, welcher den Datengebenden für seine zur Verfügung gestellten Daten, angemessen entlohnt. Dies geschieht durch eine Transaktion von dem Datennutzenden an den Datengebenden, die trotz des Austausches von Zahlungsinformationen, die Privatsphäre des Datengebenden und dessen Identität so weit wie möglich schützt. Hierfür könnte bspw. eine leicht erweiterte Version des GNU Taler Zahlungssystem verwendet werden 3.2, um den Zahlungsverkehr zwischen den Parteien zu ermöglichen. Um dieses System vor Missbrauch zu schützen, wird ein Reputationssystem eingesetzt, um zu verhindern, dass ein Datengebender wertlose Daten oder qualitativ niedrig wertige Daten, in großer Masse bereitstellen kann, um so das System auszunutzen. Es soll zum einen, gleich wie der Bezahlvorgang, die Identität des Datengebenden schützen. Gleichzeitig soll es den Datennutzenden, der die Bewertung ausstellt, davon abhalten, das System durch mehrfaches Bewerten auszunutzen. Auch Betrugsversuche des Datengebenden sind nicht zu vernachlässigen, wie bspw. die Neuanmeldung eines Nutzers mit schlechter Reputation, um diesen Wert zurückzusetzen.

Dafür wird in dieser Arbeit der aktuelle Stand der Forschung in dem vorliegenden Kontext evaluiert und geprüft was bereits anwendbar und wo noch Lücken zur gewünschten Verwendung bestehen. Mit den gesammelten Forschungsergebnissen werden daraufhin ein Konzept präsentiert, um das gerade genannte Ziel in das Treuhändermodell zu integrieren. Dieses Konzept wird des weiteren in ein bestehendes Treuhändersystem eingebaut, um konkrete Vergleichsgrundlagen zu erhalten. Bei dem hierfür vorgesehenen System, handelt es sich um das Tresor-Projekt der Universität Hamburg [Ham22].

2 Hintergrund

Die zunehmende Digitalisierung unseres Alltags ist unbestreitbar [ZDF23]. Viele der täglichen Aktivitäten hängen stark mit ihr zusammen. Sei es den schnellsten Weg zu Arbeit finden mit Google Maps, das kontaktlose Bezahlen an der Kasse mit GooglePay oder Applepay, die Benutzung von Social Media zur Unterhaltung oder der Onlinehandel über Anbieter wie Amazon. Sie alle liefern Komfort, der durch die zunehmende Verwendung von Computern ermöglicht wird, welche im Hintergrund Unmengen an Daten für ihre Berechnungen verwenden. Diese Daten stammen meist von den Benutzern selbst. Bspw. die Standortdaten für die Berechnung potenzieller Staus im Straßennetz [han17] oder die Unterhaltungsinteressen basierend auf Watchtime von bestimmten Social Media Inhalten.
Aufgrund des ständig wachsenden Markts für neue Digitaltechnologien ist auch die Nachfrage nach Daten über die letzten Jahre in Höhe gestiegen. Über das letzte Jahrzehnt haben Daten, Öl als wertvollste Ressource abgelöst [FK20]. Während 2008 die vier weltweit wertvollsten Unternehmen Ölkonzerne waren, waren 2018 bereits die 7 wertvollsten Unternehmen Internet- und Technologiefirmen.

2.1 Herkömmliche Datenkommunikation

Unter Anbetracht des hohen Wertes von Daten sind viele Unternehmen verständlicherweise sehr zurückhaltend, was den Austausch betrifft. Schließlich beutet eine eigene Sammlung von Daten ein potenzielles Verkaufsgut. Laut einer durch die Bundesregierung aufgegriffenen Studie von Fedkenhauer et al. geben zwar viele der befragten Unternehmen an, Aktivitäten im Bereich des Datenaustausches zu betreiben, allerdings umfasst das in 83% der Fälle den Austausch von Daten mit Kundinnen und Kunden. 53% der Unternehmen teilen ihre Daten mit Lieferantinnen und Lieferanten. Ein noch kleinerer Anteil von 21% teilen ihre Daten mit Unternehmen aus der gleichen oder anderen Branchen und nur 15% teilen sie mit Wettbewerbern [Bun21].

Aus der kapitalgetriebenen Sicht eines Unternehmens besitzt das Teilen der eigenen Daten keinen direkten Nutzen. Da ein Unternehmen seine eigene Gewinnmaximierung anstrebt, ist das Teilen von Daten eher ein Nachteil, da fremde Unternehmen mit den selbst gesammelten Daten ihre Produkte qualitativ erweitern können. Dadurch werden entweder anderen Wettbewerbern oder branchenfremde Unternehmen in ihrem Marktwert gefördert, was zu dem Sinken des eigenen Marktwertes führt.
Diese protektive Herangehensweise kann allerdings auch der Gewinnmaximierung im Weg stehen. Im Fall von einem direkten Tausch an Daten können beide Parteien einen Profit aus der Interaktion erwirtschaften. Die Bundesregierung selbst schreib in [Bun21] das kaum Datenkooperationen zwischen staatlichen und wirtschaftlichen Akteurinnen bestehen, obwohl die staatlich gesammelten Daten eine Grundlage für wirtschaftliche Innovation sein könnten. Im Gegenzug könnten die Daten von Unternehmen dem Staat bei der Sicherstellung seines Versorgungsauftrages, der Daseinsvorsorge und der Wahrung öffentlicher Schutzgüter helfen. Dies ist eine optimale Situation für die Verwendung eines Datentreuhänders.

2.2 Datentreuhänder

Ein Datentreuhänder ist ein neutraler vertrauenswürdiger Vermittler von Daten eines Datengebenden zu einem Datennutzenden. Er hat selbst kein kommerzielles Interesse an der Verwertung der Daten und agiert vergleichbar zu einem Notar strickt für den Datengeber. Seine Hauptaufgaben umfassen meist

die Kontrolle von Zugriffsrechten, das Kontrollieren von Einhaltung der Datenschutzrichtlinien, sowie das Verschlüsseln oder Anonymisieren von Datenbeständen. In speziellen Fällen kann ein Datentreuhänder auch die Auswertung von Daten vornehmen. [Bun21][Ric]

Da wie bereits angeführt viele Unternehmen die Weiterleitung ihrer Daten vermeiden, ist unter der Annahme eines etablierten Datentreuhänders ein deutlich größerer Datenbestand verfügbar. Bereits heute - vor einer großen Etablierung von Datentreuhänder - verspricht das Konzept einige gesellschaftliche Vorteile: [Ric]

1. Durch sie können Datenbestände besser vernetzt werden und Zusammenhänge hergestellt werden, welche zu Innovationen führen.

2. Der Wettbewerb unter Firmen wird gestärkt, da mit besser zugänglichen Daten auch kleinere Unternehmen, die kein Datenmonopol besitzen ihre Produkte aufwerten können.

3. Der individuelle Endnutzer erhält mehr Kontrolle und Transparenz über die Speicherung und Verwendung seiner Daten.

Allerdings ist das Verständnis eines Datentreuhänders nicht eindeutig. Jürgen Kühling beschreibt den Datentreuhänder als “ein schillerndes Wesen. Jeder kennt ihn, jeder setzt ganz eigene Hoffnungen in ihn – und jeder stellt sich doch etwas anderes unter ihm vor“ [KB21]. Obwohl die Technologie eines Datentreuhänders bereits seit Jahren existiert und verwendet wird [Har18] ist es nicht gelungen eine konkrete allumfassende Definition für die Technologie zu finden.

2.2.1 Definition

Die allgemeingültigste Definition stammt aus der Rechtswissenschaft und bezieht sich auf die Treuhandschaft im Allgemeinen. *[Treuhandschaften sind ein] Rechtsverhältnis, bei dem eine natürliche oder juristische Person (Treugeber) einer zweiten Person (Treuhänder) ein Recht unter der Bedingung überträgt, von diesem Recht nicht zum eigenen Vorteil Gebrauch zu machen. [...] Gemeinsames Charakteristikum ist die Uneigennützigkeit und Vertrauenswürdigkeit bei der Wahrnehmung fremder Interessen bzw. die uneigennützige Ausübung von amtlichen Befugnissen"*[Pro13]. Diese Treuhandschaft wurde in der Vergangenheit verwendet, um bspw. Ländereien zu verwalten und im Namen einer lokalen Gemeinschaft Entscheidung zu treffen. [Har18]. In dem Fall eines Datentreuhänders bedeutet dies konkret, dass eine Datengebende Person beim Bereitstellen ihrer Daten den Datentreuhänder dazu ermächtigt, über die Daten zu verfügen. Darunter fällt unter anderen auch die Weitergabe der Daten, solange die im Sinne des Datengebers ist.

2.2.2 Verschieden Modelle

Insgesamt lassen sich alle bis zum heutigen Zeitpunkt in Betrieb genommenen oder geplanten Datentreuhandsysteme wie folgt kategorisieren. Zum einen besteht die Einteilung in Customer to Business oder Business to Business Systeme und zum anderen die risikobasierte Einteilung nach Zentralen/Dezentraler Datenspeicherung und Freiwilliger/Verpflichtender Nutzung. (siehe 2.1)

Die Unterscheidung zwischen Customer to Business und Business to Business Datentreuhändern basiert ausschließlich auf den interagierenden Parteien. Im Falle einer B2B Interaktion kommunizieren zwei Unternehmen die vorhandenen Daten miteinander. In diesem Fall kommt es häufig vor, dass eines der Unternehmen durch eine staatliche Behörde dargestellt wird. Bei den gespeicherten Daten handelt es sich meist um personenbezogene Daten. Aufgrund dessen befassen sich B2B Datentreuhänder häufig mit der Pseudonymisierung und der Verwaltung der bereitgestellten Daten. Sie sind unter anderem im Gesundheitswesen viel vertreten. [Bla+20] C2B Systeme umfassen solche, bei denen einen Endnutzer die Daten generiert und diese an ein Unternehmen zur weiteren Benutzung freigibt. Ihre Aufgabe ist hauptsächlich die Unterstützung des Nutzers bei der gerechten Weiterverarbeitung seiner

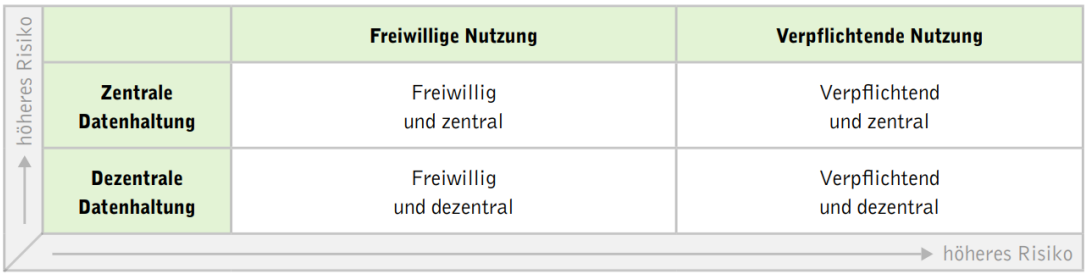

Abbildung 2.1: Risikobasierte Unterscheidung von Datentreuhandmodellen [BS21b]

Daten. [Bla+20] Hier sind dementsprechend die Pseudonymisierung der Daten sowie die Einhaltung von Datenschutzrechtlinien und einheitlicher Standards die Hauptziele des Datentreuhänders.

Des Weiteren lassen sich Datentreuhandsysteme anhand ihrer Datenspeicherung sowie Nutzung kategorisieren und Risikotechnisch bewerten. Die zentrale Speicherung der Daten bietet einige Vorteile für den Treuhänder. Sie ermöglicht es Daten vorzuverarbeiten, zu analysieren und Datenverarbeitende von dem direkten Zugang der Daten auszuschließen. Allerdings birgt die zentrale Datenspeicherung ein enormes Risiko der Datensicherheit, da hier ein Single Point of Failure entsteht. Bei einem Angriff auf einen solchen Datentreuhänder fällt es einem Angreifer somit leichter eine große Menge an Daten zu stehlen. Bei einer dezentralen Speicherung werden die Daten durch den Treuhänder pseudonymisiert oder ganz anonymisiert und bei dem Datengeber gelagert. Aus diesem Grund bietet eine dezentrale Datenspeicherung mehr Sicherheit vor Diebstahl. Sie zieht allerdings auch eine eingeschränkte Verarbeitung und Analyse der Daten mit sich und erhöht die Komplexität der Verwaltung.

Es besteht eine weitere Unterteilung in Datentreuhandsysteme, dessen Benutzung freiwillig ist oder verpflichtend ist. Dabei fällt der größte Anteil an Systeme unter die freiwillige Benutzung. Es gibt jedoch auch Szenarien, in denen die Verwendung einer Datentreuhand verpflichtend ist. Ein Beispiel hierfür wäre das Krebs- und Transplantationsregister aus dem medizinischen Bereich. Das Risiko steigt bei verpflichtenden Systemen, da sie meist einen wichtigen Bestandteil der Kommunikation ausmachen der nicht umgangen werden kann. Somit sind die potenziellen Schäden, die bei einem Angriff entstehen können, höher als in einem freiwilligen System.

2.3 Anwendungsfälle

Das mögliche Spektrum an Anwendungsfällen ist denkbar breit. In beinahe jedem Bereich, der eine große Menge an Daten benötigt oder verwaltet ist die Verwendung eines Datentreuhänders angedacht. Beispiele dafür sind:

- **Patientendaten** Der Datentreuhänder sorgt für eine Pseudonymisierung von Patientendaten zur Bereitstellung an Forschungseinrichtungen. Hierbei behält der Patient die Kontrolle über seine Daten und kann selbst entscheiden mit wem seine Daten geteilt werden.
- **Autonomes Fahren** Beim autonomen Fahren werden enorm viele Daten generiert, welche seit 2017 per Gesetz gespeichert werden müssen [Bun]. Leider ist die Zugehörigkeit der Daten rechtlich weder dem Autohersteller noch dem Autoinhaber zuzuschreiben [Ric]. An dieser Stelle kann ein Datentreuhänder die Kommunikation erleichtern und exklusiven Zugang von bspw. Versicherungen oder Automobilkonzernen ausschließen.
- **E-Government** Durch die Verwendung eines Datentreuhänders können bei der behördlichen Verwaltung von Bürgerdaten große Fortschritte erzielt werden. Unter anderem müssen so notwendige Informationen aus anderen Registern für Verwaltungsvorgänge nicht mehr vom Bürger bereitgestellt werden. Der Bürger gibt lediglich seine Einwilligung zum Abruf der Daten zu Beginn

des Verwaltungsvorgangs. Auf diese Weise können Verwaltungsvorgänge erheblich effizienter Ablaufen.

- **KI-Datenpools** Eines der größten Probleme bei Entwicklung von KI-Software ist der Zugang zu einer ausreichend großen Menge an Trainingsdaten. Ein Datentreuhänder kann solche Daten, die zur Verwendung für KI-Training freigegeben wurden, sammeln und pseudonymisiert an mehrere Interessierte verteilen. Dadurch entsteht ein einheitlicher Zugang zu Trainingsdaten der gleichzeitig nur freigegebene Daten beinhaltet und so rechtlichen Streit über das Copyright aus dem Weg geht.

- **Industrie** In der Industrie besteht eine hohe Abhängigkeit von Warenbewegungen, seien es in Lieferketten, Logistik oder Handel. Durch die Verwendung eines Datentreuhänders können diese Informationen pseudonymisiert an Warenempfänger weitergegeben werden. Vor allem in diesem Bereich besteht durch die Zusammentragung an Lieferinformationen in Kombination mit Algorithmen der Graphentheorie ein großes Innovationspotential.

- **PIMS** Personal Information Management Systeme befassen sich grundsätzlich mit der Wahrung von personenbezogenen Daten und bietet ihren Nutzern mehr Kontrolle über diese. Datentreuhänder sind für solche Systeme vor allem von Vorteil da sie im Umgang mit personenbezogenen Daten dem Nutzer wieder die Kontrollen über seine Daten zurückgeben.

[BS21a][BS21b][Bun]

Der erste Datentreuhänder entstand einer bereits im Jahre 2006 in England. Seitdem das Thema Datentreuhänder in den letzten Jahren immer mehr zum Trend wurde [Ric]. Die “UK Biobank“ ist eine biomedizinische Datenbank die sowohl medizinische Daten als auch biologische Proben von einer halben Millionen Teilnehmer aus Großbritannien speichert[Har18]. Die gespeicherten Daten sind pseudonymisiert und werden ausschließlich an Forscher im Feld der Medizin weitergegeben, was die UK Biobank zu einem Paradebeispiel für einen Patientendatentreuhänder macht.

2.4 Bestehende Anreize

Generell sei gesagt, dass die Verwendung eines Datentreuhänders keine direkten Nachteile mit sich bringt. Wenn Daten ohnehin geteilt werden oder werden müssen, so behält der Nutzer bei der Speicherung der Daten über einen Datentreuhänder mehr Kontrolle über die Verwendung seiner geteilten Daten, als im Vergleich mit dem direkten Teilen mit Unternehmen. Durch den Datentreuhänder wird ihm ermöglicht zu einem beliebigen Zeitpunkt das weitere Teilen seiner Informationen einzustellen. Vermutlich teilen deswegen Privatpersonen ihre Daten lieber mit Datentreuhändern als auf direktem Weg. [TRE24]

Allerdings gibt es nur wenige Vorteile, die die freiwillige Benutzung eines Datentreuhänders reizvoll machen. Bei freiwilligen C2B Datentreuhändern, hängt die Entscheidung für oder gegen die Nutzung des Treuhänders beim Nutzer. Es ist also an ihm abzuwägen, ob die Verwendung ausreichend Vorteile liefert. Im Fall von PIMS-Treuhändern wird die Verwendung von manchen als erstrebenswert angesehen, da der Nutzer mehr Kontrolle über die Verbreitung von persönlichen Daten erhält und selbst entscheiden kann, mit wem diese geteilt werden. Die Erkenntnisse von Jai et al. [JK16] zeigen hingegen, dass vor allem jüngere Erwachsene weniger Wert auf den Schutz ihrer persönlichen Daten legen.

Direkte Anreize zur Verwendung einer solchen Software, sind bisher kaum Präsent. An der Weitergabe der persönlichen Daten, hat nur das Business, dass einen konkreten Mehrwert. Der Nutzer, der seine Daten freigibt, erhält keine Kompensation in irgendeiner Form. Folglich kann es für einen freiwilligen Datentreuhänder mühsam sein, neue Nutzer zu gewinnen und die Technologie als solche auszubauen.

3 Anforderungen und verwandte Arbeiten

3.1 Anforderungen

Da eine Lösung zur Verwendung durch einen Datentreuhänder konzeptioniert ist, decken sich die Anforderungen an eine mögliche Lösung stark mit denen eines üblichen Datentreuhänders. Im Grunde kann zwischen funktionalen und nichtfunktionalen Anforderungen unterschieden werden.

3.1.1 Funktionale Anforderungen

1. Ein Bezahlsystem ermöglicht es einem Datengebenden Geld für seine Daten zu erhalten.
2. Die Präsenz eines Reputationswertes gibt den Datennutzenden vor Erwerb der Daten eine Einschätzung über deren Qualität.
3. Nach Abschluss der Transaktion kann ein Datennutzender den entsprechenden Datengebenden aufgrund der Qualität der übermittelten Daten bewerten.
4. Ein Datengebender muss eine Bewertung seiner Daten ermöglichen.
5. Ein Datennutzender kann pro Austausch nur genau eine Bewertung für einen Datengebenden abgeben. Mehrfache Bewertungen ist nur im Fall von mehrfachem Erwerb möglich.

3.1.2 Nicht funktionale Anforderungen

1. **Anonymität** Die Identität des Datengebenden darf durch den Austausch von Zahlungsmitteln oder durch dessen Reputation nicht nachverfolgbar sein.
2. **Zeitsensitivität** Der Bezahlvorgang muss in vernachlässigbarer Zeit geschehen.
3. **Skalierbarkeit** Die Rechenzeit des Systems soll bei linear steigender Menge an Datengebenden und Datennutzenden auch mit linearem Zeitaufwand zunehmen.
4. **Vertraulichkeit** Die kommunizierten Daten dürfen nicht durch unbefugte Dritte ausgelesen werden können.
5. **Integrität** Die kommunizierten Daten dürfen nicht unbemerkt durch unbefugte Dritte verändert werden.

3.2 GNU Taler

Das im Paper “Enabling Secure Web Payments with GNU Taler“ von J. Burdges et al. eingeführte Zahlungssystem GNU Taler, ist ein elektronisches Online-Zahlungssystem, das Datenschutz für Customer und Mechanismen zur steuerlichen Nachverfolgung für Merchants bietet [Bur+16]. Es verwendet einen Exchangeservice, um Münzen mithilfe von blinden Signaturen zwischen Nutzern und Händlern zu transferieren. Im Folgenden werden diese Münzen als Taler bezeichnet, um Verwirrung zu vermeiden. Das System basiert auf 4 übergeordneten Rollen, dessen Interaktion grob in Abbildung 3.1 skizziert ist. Der Customer möchte ein Gut oder eine Dienstleistung bei dem Merchant erwerben und bezahlt diesen dafür mit Talern, welcher er beim Exchange erworben hat. Der Merchant kann die erhaltenen Taler

wieder beim Exchange für herkömmliche Währungen eintauschen. Ein Auditor überprüft währenddessen die Liquidität des Exchange, um sicherzustellen, dass dieser auch bei Datenverlust von Talern noch in der Lage ist allen Beteiligten, Auszahlungen zu ermöglichen.

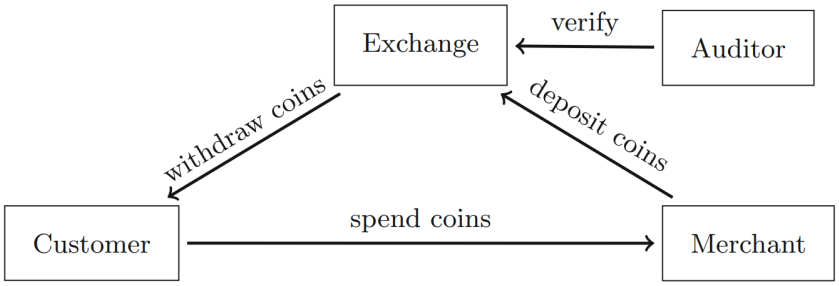

Abbildung 3.1: Grundlegender Ablauf des GNU Taler Systems [Bur+16]

Taler abheben. Damit ein Customer Geld auf sein Wallet laden kann, muss er sich zuerst bei seiner Bank anmelden. Sollte die Bank GNU Taler native unterstützen, so kann der Customer in einem Formular eine Summe auswählen, die er in Taler übertragen möchte und einen Exchangeservice, über welchen der Tausch abgewickelt wird. Nachdem der Customer die Transaktion bestätigt hat, wird die ausgewählte Summe transferiert, der Exchangeservice signiert die äquivalente Summe an Talern und überträgt diese in das Wallet des Customers.

Taler ausgeben. Gehen wir von der Situation aus, dass eine Customer bei einem Merchant (hier ein Onlineshop), etwas erwerben möchte. Nach der Auswahl des Produktes und GNU Taler als Zahlungsmittel, erstellt der Merchant einen Zahlungsvertrag, der Details wie den Gesamtpreis, mögliche offene Umwandlungsgebühren und akzeptierte Exchangeservices beinhaltet und sendet diesen an das Wallet des Customers. Wenn der Customer daraufhin die Zahlung übermittelt, so leitet der Merchant die erhaltenen Taler direkt an den Exchange weiter. Wenn der Exchange den Eingang bestätigt, so kann der Merchant dem Customer die Transaktion bestätigen und der Kauf ist somit abgeschlossen.

In dem Treuhändermodell eignet sich GNU Taler aber nur teilweise als Zahlungssystem. Der Datengebende stellt dabei den Merchant da, der seine Daten als digitales Gut anbietet. Der Datennutzende stellt hier die Rolle des Customers dar und möchte diese Daten erwerben. Sollte der Datengebende der Anfrage des Datennutzende zustimmen, so würde er einen Zahlungsvertrag formulieren, um den Anspruch auf seine Vergütung zu formalisieren. Allerdings soll der Datengebende im Datentreuhändermodell so gut wie möglich vor Informationsgewinnung geschützt werden. Bei dem von GNU Taler vorgeschlagenen Prozess wird jedoch die Identität des Datengebenden nachverfolgbar, während der Datennutzenden anonym bleibt. Zusätzlich besteht eine direkte Kommunikation zwischen Datengebenden und Datennutzenden, was weitere schützenswert Information über die Identität des Datengebenden (bspw. seine IP-Adresse) preisgibt.

3.3 Bitcoin

Die Idee hinter Bitcoin entstand 2008 durch eine Person oder Gruppe mit dem Namen Satoshi Nakamoto. Heute ist Bitcoin die mit Abstand weit verbreiteteste Onlinewährung weltweit [Sch24]. Sie basiert auf einem verteilten öffentlichen Register, das alle Transaktionen pseudonymisiert für jeden einsehbar macht und speichert. Dieses Register ist auch als Blockchain bekannt.

Die Blockchain ist eine Kette an Blöcken, die jeweils den Hash des vorherigen Blocks und neue Transaktionen speichert. Durch den Hash des vorherigen Blocks entsteht eine Kette, in der jeder Block Informationen über seinen Vorgänger speichert. Ein neuer Block entsteht durch einen sogenannten Proof of Work. Dieser ist das Wissen über eine Zufallszahl, deren Hash mit x Null bits beginnt, welche

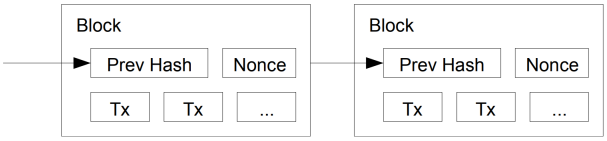

Abbildung 3.2: Blockchain mit Transaktionen [Nak08]

ausschließlich durch das wiederholte Ausprobieren von Zufallszahlen gefunden werden kann. Dieser Vorgang wird als Bitcoin Mining bezeichnet. Sobald eine solche Zahl gefunden wird, kann der aktuelle Block abgeschlossen werden und alle folgenden Transaktionen werden in dem nächsten Block gespeichert [Nak08].

Der Algorithmus, der bestimmt wie schwer das Erbringen des Proof of Works ist, kann jederzeit dynamisch an die Rechenleistung der Miner angepasst werden, so dass im Schnitt alle 10 Minuten ein neuer Block gefunden wird [Sch24]. Um also eine vergangene Transaktion auf der Blockchain zu verändern, müsste ein Angreifer für jeden darauffolgenden Block selbst einen proof of work berechnen. Ein Angreifer kann deswegen nur eine vergangene Transaktion ändern, wenn er allein mehr Rechenleistung besitzt als alle anderen Miner zusammen.

In dem Moment in dem ein Block abgeschlossen und der nächste begonnen wird, sind alle Transaktionen auf dem abgeschlossenen Block ein Mal bestätigt. Das bedeutet das keine der Transaktionen einen Coin zum zweiten Mal ausgibt. Dies ist eine essenzielle Sicherheitsmaßnahme von Blockchain Kryptowährungen da ohne Blockchain dezentralisiert ohne neutralen Überprüfer funktioniert. Es ist also an alle Beteiligten die pseudonymisierten Transaktionen des Blocks zu überprüfen. Sollte ein Angreifer die Zufallszahl finden und vor Abschluss des Blocks noch eine doppelte Ausgabe hinzufügen so kann diese Transaktion erst festgestellt werden, wenn bereits neue Transaktionen in dem nächste Block hinzugefügt werden was zeigt, dass nur weil eine Transaktion bestätigt ist, sie nicht direkt vertrauenswürdig ist. Im Fall von Bitcoin wird empfohlen 6 weitere Blöcke abzuwarten, bis eine Transaktion wirklich abgeschlossen ist []. In Kombination mit einer durschnittlichen Berechnungsdauert von 10 Minuten dauert es also ca. eine Stunde, bis eine Transaktion auf der Blockchain als Sicher angesehen wird. Durch den enormen Rechenaufwand der benötigt wird, um einen Block abzuschließen, kann grob überschlagen werden wie viel Energie in eine einzelne Bitcointransaktion fließt. In [Dig23] wird eine Transaktion auf 703,25 kWh geschätzt, was ca 470.000 VISA Transaktionen entspricht.

Allein die lange Bestätigungsdauer einer Bitcointransaktion schließt es bereits für die Verwendung in diesem System aus, da eine Stunde nicht mehr als vernachlässigbare Zeit angesehen werden kann und somit nicht Zeitsensitiv ist. Hinzu kommt der gigantische Stromverbrauch, welcher die Skalierbarkeit von Bitcoin selbst in Frage stellt.

3.4 Ethereum

Ethereum ist die Blockchainanwendung hinter der zweit meist verbreitetesten Kryptowährungen Ether [Coi]. Ein gängiger Irrtum ist jedoch, dass Ethereum selbst keine Währung ist. Ethereum wurde 2013 von Vitalik Buterin erfunden und ist eine Plattform, die es ermöglicht Entwicklern Anwendung auf der Blockchain zu entwickeln. Es liefert ein Turing-vollständige Programmiersprache die es Entwicklern erlaub eigene Währungen in unter 20 Zeilen Code zu schreiben [But+13]. Sämtlich dort geschrieben Währungen oder sogenannte Smart Contracts werden über die Ethereum Blockchain pseudonymisiert öffentlich zur Verfügung gestellt.

Durch die Verwendung der Blockchain decken sich einige Eigenschaften mit Bitcoin. Es besteht genauso aus einer Reihe an Blöcken, welche kontinuierlich ihren Vorgänger referenzieren. Bis 2022 nutzte Ethereum ebenfalls einen Proof of Work Ansatz. Doch seit 2022 basiert Ethereum einem sogenannten Proof of Stake Konzept [TEA24]. Das bedeutet, dass es eine Gruppe an Validierern

gibt, welche die Transaktionen innerhalb der Blöcke überprüfen. Um ein Validierer zu werden, muss ein Starteinsatz von 32 Ether gezahlt werden. Alternativ kann sich ein Nutzer einem Validiererpool anschließen und einen kleineren Starteinsatz zahlen. Dafür muss er jedoch die erzielten Gewinne teilen.

Die Aufgabe eines Validierers ist es unzulässige Transaktionen festzustellen. Im Fall eines Angriffs auf einen Block, wird der Block von Gasper (Einer Mischung des Casper-FFG Protokolls und LMD Ghost Algorithmus [But+20].) markiert. Anschließend entscheiden die Validierer ob der Block zugelassen oder blockiert werden soll. Validierer die sich bösartig Verhalten werden dadurch bestraft, dass ihr Starteinsatz nach und nach “verbrannt“ wird. Mit verbrannt ist gemeint, dass der Einsatz an ein Wallet ohne private key gesendet wird, was die Coins unwiderruflich unzugänglich macht. Durch diesen alternativen Ansatz kann der große Rechenaufwand von Bitcoin umgangen werden.

Die Verwendung einer Blockchain stellt hier wieder das Problem der Zeitsensitivität. Zwar dauert das Erstellen eines Blocks bei Ethereum nur 12 Sekunden [YCH]. Dafür wird allerdings eine Mindestanzahl von 30 Blocks empfohlen, bevor eine Transaktion für gültig erklärt wird. Somit entsteht eine Wartezeit von ca 6 Minuten, was bei weitem besser ist als Bitcoin. Trotzdem sind 6 Minuten keine vernachlässigbare Zeit, weshalb auch Ethereum nicht für dieses System verwendet werden kann.

3.5 Privacy Pass

Ein VPN bietet einige Vorteile in der regulären Kommunikation über das Internet. Bspw. erhöht er die Anonymität des Nutzers, da dessen private IP-Adresse so geschützt bleibt. Allerdings gibt es auch Nachteile die häufig übersehen werden. Einer davon ergibt sich daraus, dass wenn der Internetverkehr von hunderten Benutzern des VPNs über die gleiche IP-Adresse verteilt wird, dann genügt ein bösartiger Nutzer, der die IP-Adresse für einen Angriff oder o.ä. nutzt, um der IP einen schlechten Ruf bei populären Content Delivery Networks wie Cloudflare zu verschaffen. Ein Content Delivery Network (kurz CDN) ist dafür zuständig häufig angefragt Webseiten wie Google.de oder Netflix.com in seinem Cache aufzubewahren und so die Zugriffszeit, welche durch physikalisch große Distanzen zwischen Nutzer und Server entsteht zu verkürzen [aka]. Zusätzlich liefert ein CDN eine Menge an Sicherheitsfunktionen, wie unter anderem IP-Adressen mit schlechtem Ruf ein CAPTCHA präsentieren, um Botzugriffe zu verhindern. Daraus resultiert, dass ein regulärer Nutzer eines VPNs erheblich mehr CAPTCHAs lösen muss als ein Nutzer, der keinen VPN verwendet.

3.5.1 Umgehen vom CAPTCHAs

Privacy Pass ist eine Browsererweiterung, die es ermöglicht, vorab ein CAPTCHA zu lösen und damit eine Menge an Token zu erhalten. Solange ein Nutzer über mindestens einen Token verfügt, kann er das nächste Mal, wenn ein CDN ihn aufgrund eines schlechten IP-Rufwertes zum Lösen eines CAPTCHAs auffordert, anstatt einen Token einlösen und kann die Aufgabe so überspringen. Dadurch erhöht sich die Nutzerfreundlichkeit unter der Verwendung eines VPNs da die Anzahl an zu lösenden CAPTCHAs rapide sinkt. [Dav+18]

3.5.2 Funktionsweise

Das System ist in eine sogenannte Signierphase und Einlösephase aufgeteilt. Die Signierphase startet, nachdem der Nutzer erfolgreich ein CAPTCHA gelöst hat. Sie ist dafür zuständig dem Nutzer eine Anzahl an Token auszustellen die durch den Server signiert sind. Die Einlösephase beginnt, wenn ein CDN ein CAPTCHA für den Zugang zu einem Webinhalt fordert. Bei ihr wird einer der gespeicherte signierte Token eingetauscht, um die Lösung des CAPTCHAs zu überspringen.

Signierphase Erstellen und signieren von Token

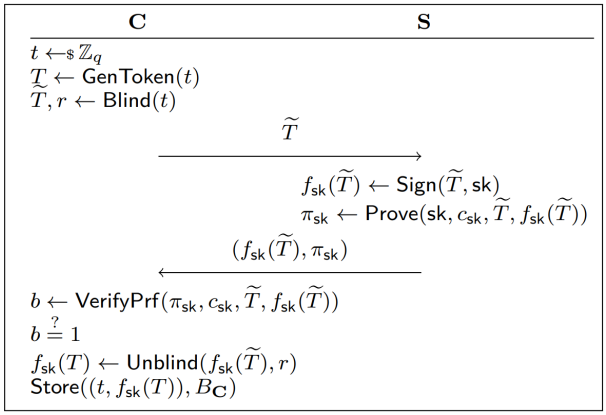

Abbildung 3.3: Signierphase von Privacy Pass [Dav+18]

Zuerst erstellt der Nutzer *C* (Client) einen Tokenseed t mit $t \in_R \mathbb{Z}_q$. Daraus generiert er Token T bspw. mit einer Hashfunktion und blindet diesen mit r wie in 3.7 beschrieben, um $\widetilde{T}$ zu erhalten. Dieser geblindete Token wird nun an den Server *S* gesendet, damit dieser in signieren kann. Beim Server angekommen beginnt dieser damit, den Token mit seinem privaten Schlüsselt sk zu signieren. Anschließend erstellt er einen sogenannten Batch Discrete Log Equivalence Proof (BDLEQ), um dem Nutzer zu beweisen, dass er für jeden Nutzer einen gleichen privaten Schlüssel verwendet. Sollte er für jeden Nutzer einen eigenen private Schlüssel benutzen, so kann er den Nutzer über längere Zeit deanonymisieren, da er die in der Signierphase erhaltenen Token mit denen der Einlösephase verlinken kann. Der signierte Token und der BDLEQ werden wieder an den Nutzer gesendet, dieser prüft die Korrektheit des Beweises, unblindet den signierten Token und speichert ihn für spätere Verwendung.

Einlösephase Signierten Token einlösen

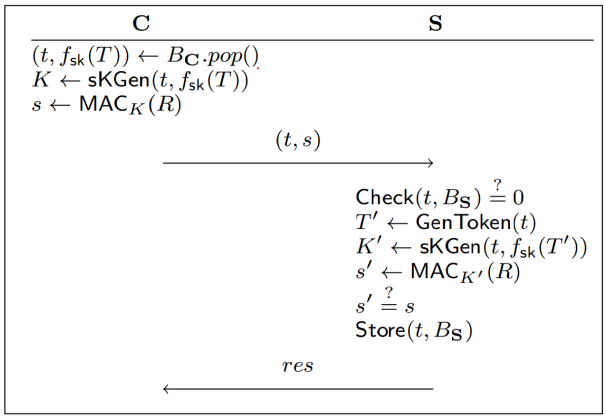

Abbildung 3.4: Einlösephase von Privacy Pass [Dav+18]

Der Nutzer beginnt damit einen gespeicherten signierten Token auszulesen und einen anfragenabhängigen Wert R zu berechnen. Dieser könnte einfach die Domain der Anfrage sein. Er generiert einen shared Key K aus dem Tokenseed und dem signierten Token und verschlüsselt R mit K als Key. Daraufhin sendet er das Tupel aus Tokenseed t und $s \leftarrow MAC_K(R)$ mit shared Key verschlüsseltem R an den Server. Der Server prüft, ob t bereits für eine vorherige Anfrage verwendet wurde. Falls dies nicht der Fall ist, berechnet er auf Grundlage von t alle Schritte des Nutzers und erhält so ein s'. Sollte $s' = s$ gelten, dann ist der Token valide, das CAPTCHA kann übersprungen werden und der Nutzer erhält Zugriff auf die angefragt Webressource.

Batch Discrete Log Equivalence Proof Beim dem BDLEQ handelt es sich um einen zero knowledge proof. Zero knowledge proofs sind ein in der Kryptographie auftretendes Beweismuster, dass das Wissen über einen Wert belegen kann ohne den Wert tatsächlich zu nennen. Der hierverwendete Discrete Log Equivalence Proof kann wie sich aus dem Namen ableiten lässt dazu verwendet werden, die Gleichheit von zwei diskreten Logarithmen zu zeigen. Für die genau Anwendung wird hier auf die Arbeit von Davidson et al. [Dav+18] verwiesen. Dieser Beweis wird aus dem Grund erstellt, dass ein CDN ansonsten für das Signieren von jedem Token ein unterschiedliches Schlüsselpaar verwenden kann. Die Folgenden davon sind, eine mögliche Deanonynmisierung, da der CDN beim Einlösen des Tokens anhand des verwenden Signierschlüssels der einmalig ist, den Signierzeitpunkt und zugehörigen Nutzer bestimmen kann. Der DLEQ liefert dem Nutzer hier die Sicherheit, dass der diskrete Logarithmus des zum signieren verwendeten private Keys der gleiche ist, wie ein von CDN öffentlich bekannt gegebener diskrete Logarithmus. Davidson et al. beschreibt zudem einen Weg den DLEQ in einer Sammelform zu formulieren, sodass ein Beweis für eine Liste an Token gilt [Dav+18], was Rechenaufwand spart.

Im Grund lässt Privacy Pass gut auf die Verwendung im Datentreuhändermodell anwenden. Unter den Annahmen, dass ein Datennutzender hier ein Nutzer ist, kann er Token bei dem Datentreuhänder erwerben, speichern und zu einem späteren Zeitpunkt wieder ausgeben. Da die Token blind signiert werden wird vermieden, dass der Datentreuhänder, die in den unterschiedlichen Phasen verwendeten Token zueinander verlinken kann und Zusammenhänge zwischen Einlösen und Ausgeben ziehen kann. Der Datentreuhänder kann eine weitere überprüfende Instanz verwenden wie in [Dav+18] beschrieben wird, um den Aufwand der Überprüfung auszulagern.
Einige Punkte sprechen jedoch gegen die direkte Verwendung von Privacy Pass.

1. Die Token haben keinen Wert. Bei Bezahlungsmittel ist es essenziell einen bestimmten Preis bezahlen zu können. Dafür wären entweder mehrere unterschiedliche Tokens von Nöten oder Token welche einen monetäre Wert gespeichert haben. Bei der Speicherung eines monetären Wertes erbringt das Problem der Beweisbarkeit. Wenn ein Token geblindet ist und für den Datentreuhänder nicht lesbar ist, wie kann sichergestellt werden, dass der richtige Wert eingehalten wurde und der Datennutzende sich nicht mehr Geld aufschreibt als ihm zusteht. Ansonsten kann nur in Vielfachen von dem einen Token gezahlt werden was dazu führt, dass es keine präzise Preisvergabe gibt oder bei jeder Transaktion eine große Menge an Token validiert werden müssen.

2. Ein Nutzer interagiert sowohl zum Einlösen als auch zum Ausgeben nur mit dem Server und nicht mit einem Verkäufer, dem er seine Token im Tausch anbietet. Es existiert hier also kein 3. Akteur wie bei GNU Taler 3.2 der erhaltene Token einlöst.

3. Die Token können ähnlich wie bei GNU Taler 3.2 nicht direkt an den Verkäufer weitergegeben werden, da dadurch eine direkte Verbindung zwischen Datennutzenden und Datengebenden entsteht, welche schützenswerte Informationen offen legt.

3.6 Privacy-Preserving Reputation Management

In ihrem Paper beschreiben R Petrlic et al. ein Reputationssystem, um Nutzern den Dienst eines Dienstleisters anonym bewerten zu lassen und so anderen Nutzern eine Einschätzung über die Qualität der Dienstleistung zu geben [PLS14]. Es werden 3 Rollen charakterisiert. Ein Reputation Provider RP, der sich um das Verwalten der verschiedenen Reputationswerte kümmert. Eine Menge an Service Providern SP, welche einen Dienst anbieten, der durch Nutzer bewertet werden soll. Und zuletzt eine Menge an Nutzern U, die die Dienstleistung der Service Provider bewerten, wobei keiner der Nutzer gleichzeitig ein Service Provider SP sein kann, $U \cap SP = \varnothing$.

Der im Paper beschriebene Ablauf des Bewertens, eines $sp \in SP$ durch einen Nutzer $u \in U$, umfasst grob die Erstellung eines Schemas für einen Bewertungsvektors durch sp, welcher von u später

verwendet wird, um den sp zu bewerten. Nach dem sp dieser Vektor mit dem RP kommuniziert wurde, kann u eine Dienstleistung von sp in Anspruch nehmen. Dieser antwortet, zusätzlich zur Erbringung der Dienstleistung, mit einer Beschreibung des Bewertungsvektors, einem Schlüsselpaar und einem Token. Dank der Beschreibung kann der Nutzer selbst einen Bewertungsvektor erstellen, in welchen er seine Bewertung der Dienstleistung einbaut. Im Anschluss wird dieser mit dem Schlüsselpaar verschlüsselt, signiert und zusammen mit dem signierten Token, an den RP gesendet. Der Token dient hier zur Erkennung von doppelten Bewertungen.

Die Berechnung der Reputation ist in Zeitslots eingeteilt, damit ein sp nicht in der Lage ist einen alten Reputationswert anzugeben. Wenn nun ein Nutzer u den Reputationswert eines sp's einsehen möchte und der Wert für den gewünschten Zeitraum noch nicht bestimmt wurde, so muss dieser zwischen RP und sp berechnet werden. Hierfür bestimmt RP die Summe aller verschlüsselten Bewertungsvektoren für den Zeitraum und sendet diese, zusammen mit weiteren Prüfwerten an sp. Dieser kann die übermittelten Werte prüfen und auf eine Blacklist hinzufügen, um Replay Attacken von RP auszuschließen. Wenn die Überprüfung gelingt, signiert sp die Summe der Vektoren zusammen mit einer ID und verifiziert somit den neuen Reputationswert an RP.

Bei einer Anfrage des Reputationswertes von sp durch u, schickt der sp eine Reihe an Werte zu u. Diese erlauben es u, den Bewertungsvektor zu interpretieren und zu prüfen, dass der übermittelte Werte sowohl aktuell, als auch nicht beeinflusst wurde. Das Konzept von Petrlic et al. bietet unter anderem Schutz vor:

1. **Whitewashing**, bei welchem sich ein sp mit schlechter Reputation als neuer sp ausgeben kann, um somit den Wert zurücksetzen kann.

2. **Transaction-independent Ratings**, bei welchen ein Nutzer die Dienstleistung bewerten kann, obwohl er besagt Dienstleistung nicht in Anspruch genommen hat.

3. **Sybil Attacks**. Ein Nutzer bewertet eine Dienstleistung unter mehreren Identitäten und täuscht so seine Meinung als Gruppenmeinung vor.

4. **Delta Analysis**. Eine teilweise Deanonymisierung des Nutzers durch vergleichen von gesammelten Bewertungen in unterschiedlichen Zeitabständen.

In der vorliegenden Situation kann das Konzept zu großen Teilen verwendet werden. Die Rollenverteilung bleibt unter den Zuweisungen von Service Provider zu Datengebenden, Reputation Provider zu Datentreuhänder und User zu Datennutzenden bestehen. Allerdings ist das Konzept darauf ausgelegt, dass zu Beginn eine Kommunikation zwischen dem Datengebenden und Datennutzenden besteht, so dass der Reputationswert direkt durch den Datennutzenden abgefragt werden kann. An dieser Stelle muss das Konzept etwas abgewandelt werden, zu keinem Zeitpunkt während der Durchführung eine direkte Kommunikation bestehen soll. Stattdessen soll der Austausch des Reputationswertes direkt über den Treuhänder ablaufen.

3.7 Blinde Signaturen

Das von David Chaum im Jahre 1983 veröffentlichtes Paper “Blind signatures for untraceable payments“ beschreibt den theoretischen Ansatz, dass ein Nutzer eine verifizierbare Signatur für eine Nachricht erhält, ohne dass der Unterzeichner den Inhalt der Nachricht kennt. [Cha83]. Ein Beispiel hierfür wäre ein Zahlungsdienstleister, der eine neuen Coin in dem Umlauf bringen möchte. Dafür sendet er seinen Reputationswert und den unkenntlich gemachten Coin an eine Zentrale Institution. Wenn diese den Reputationswert als hoch genug ansieht und somit den Zahlungsdienstleister als vertrauenswürdig erkennt, so kann sie den Coin blind signieren und an den Dienstleister zurückschicken. Wenn nun ein späterer Inhaber des Coins, prüfen möchte, ob sein Coin von einem vertrauenswürdigen Dienstleister erstellt wurde, so kann er die blinde Signatur mit dem öffentlichen Schlüssel der Institution entschlüsseln. Das Ergebnis dieser Berechnung zeigt, dass die Institution den Coin tatsächlich signiert hat und sie

dem Coinaussteller vertraut. Der hierbei essenzielle Punkt ist, dass diese Institution nicht weiß, was sie unterschreibt. Somit kann die Institution die Beziehung des Dienstleisters und des Coininhabers nicht nachverfolgen.

Die für das Verfahren nötigen Rechenschritte sind im Folgenden beschrieben. Angenommen der Unterzeichnende verfügt über eine private Signierfunktion s' und eine öffentliche Funktion s, sodass $s(s'(x)) = x$. Der Nutzer verfügt über die privaten Funktionen c und dessen invers c', sodass $c'(s'(c(x))) = s'(x)$.

1. Der Nutzer beginnt nun damit, sich ein x auszusuchen. Dieses wird durch zufällige Redundanz vor Kollisionen geschützt und mit $c(x)$ unkenntlich gemacht.

2. Anschließend erhält der Unterzeichnende $c(x)$, berechnet $s'(c(x))$ und schickt den entsprechenden Wert an den Nutzer zurück.

3. Wenn der Nutzer nun $c'(s'(c(x))) = s'(x)$ berechnet, so erhält er den signierten Ursprungswert x ohne, dass der Unterzeichnende x je kannte.

Daraufhin kann jede weitere Person die Unterschrift überprüfen, indem diese die öffentliche Funktion s verwenden um $s(s'(x))$ zu berechnen und das Ergebnis mit x abgleichen.

Blinde Signaturen sind bei dem Ansatz von privatsphäreschützenden Zahlungs- und Reputationssystemen einer der Kernbausteine der verwendet wird, um einen vertrauenswürdigen Austausch zwischen den Parteien zu ermöglichen. Sie liefern die Grundlage der Kommunikation.

3.8 Partiell Blinde Signaturen

Partiell blinde Signaturen ähneln sich vom Effekt stark zu blinden Signaturen von Chaum aus 3.7. Der jedoch entscheidende Unterschied ist, dass es bei partiell blinden Signaturen einen Infowert gibt der sowohl Nutzer als auch Signierende bekannt ist. Dieser kann genutzt werden, damit der Signierende möglicherweise Prüfung ausführen kann und zu entscheiden, ob er die Nachricht wirklich signieren möchte. Diese Eigenschaft wird im späteren Verlauf bspw. dazu verwendet einen Coin partiell blind zu signieren. Hierbei ist es essenziell, dass der Signierende den monetären Wert des Coins prüfen kann, da ein Nutzer sonst freie Kontrolle über den Wert seines eigenen Coins hat. Mit partiell blinden Signaturen, kann genau dies gewährleistet werden. Dabei ist der monetäre Wert des Coins die Info und der kryptographische Wert des Coins die Nachricht. Nun kann der Signierende prüfen, dass der monetäre Wert der vereinbarte Wert ist und kann den kryptographischen Wert des Coins blind signieren. Der springende Punkt ist der Zusammenhang zwischen Info und Nachricht, da sonst der monetäre Wert des Coins nach dem Signieren durch den Nutzer verändert werden könnte. Um dies zu verhindern, verwendet das Verfahren den Hash der Info als Teil der Signatur, so dass diese nur gültig ist, solange die Info unverändert bleibt.

3.8.1 Funktionweise

Die Durchführung einer partiell blinden Signatur besteht aus 4 Schritten. Diese 4 Schritte sind notwendig, um die Abhängigkeit von dem msg und Info herzustellen. Es wird davon ausgegangen, dass vor dem ersten Schritt beide Parteien über den Infowert verfügen. Der Signierende bildet zuerst den Hashwert der Info signiert diesen und schickt ihn an den Nutzer. Dieser bildet den gleichen Hashwert der Info und verwendet die erhaltenen Werte zusammen mit einer Reihe an Zufallszahlen, um α und β zu berechnen. ε ist der darauffolgende Hash von α, β, z und msg. Hier fließt die bereits signierte Info über α und β mit ein. Kurz darauf empfängt der Signierende e, welches nun sowohl Info als auch msg beinhaltet. Dieses e kann er nun verwenden, um Info und msg zu signieren und die aus den Berechnungen resultierende Werte an den Nutzer zu senden. Der Nutzer kann dadurch $\rho, \omega, \sigma, \delta$ zu bestimmen und zu speichern. Möchte der Nutzer zu einem beliebigen Zeitpunkt die Signatur auf ihre Gültigkeit prüfen, so kann er $\omega + \delta$ mit $H(g^{\rho} y^{\omega} || g^{\sigma} z^{\delta} || z || msg)$ vergleichen. Wenn die Werte übereinstimmen, ist die Signatur valide.

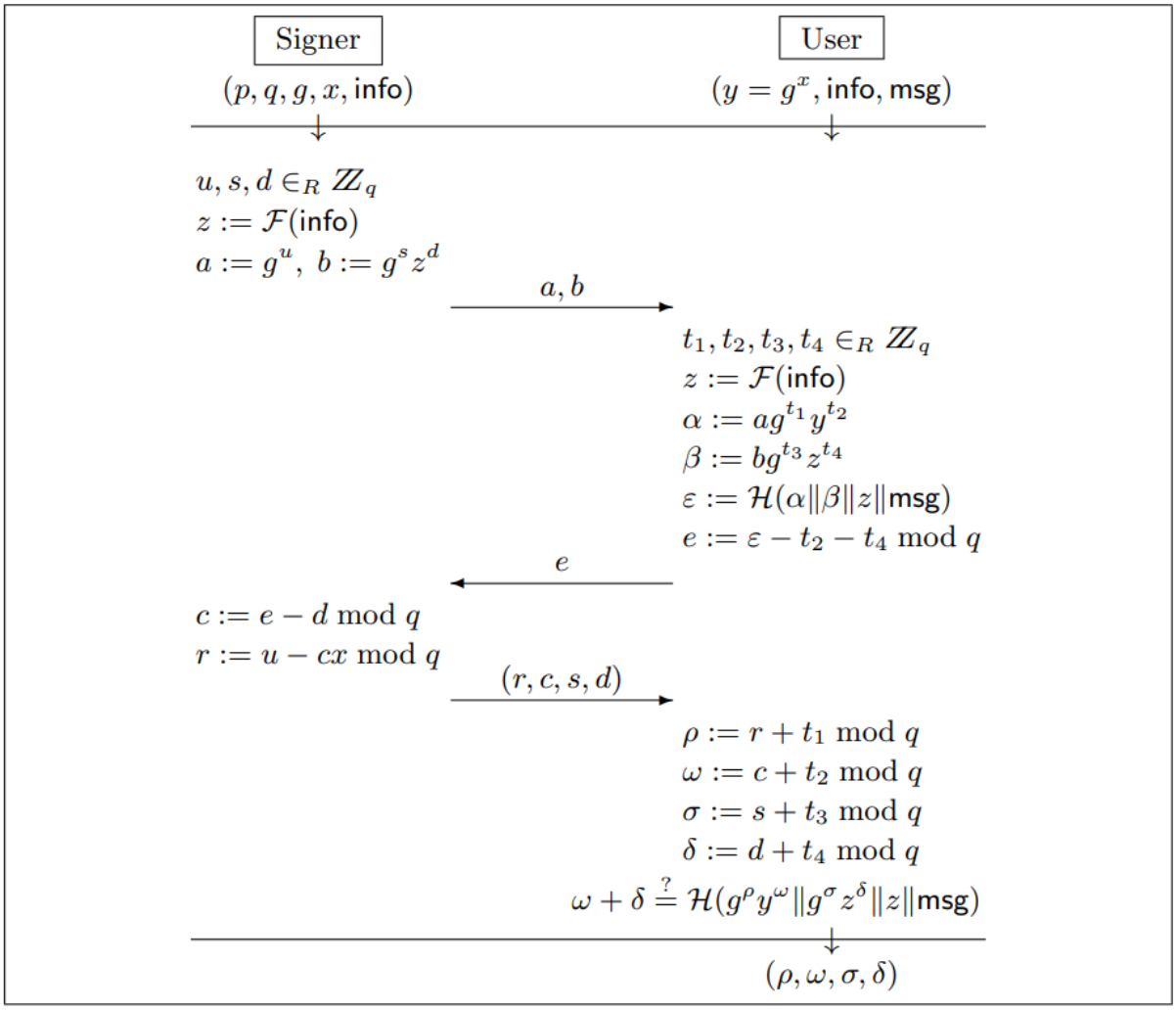

Abbildung 3.5: Ablauf einer partiell blinden Signatur [AO00]

Durch die Verwendung von Info als Teil der Signatur ist gewährleistet, dass die Signatur nur bei einer unveränderten Info und msg valide bleibt. Denn sollte sich die Info oder msg ändern, dann würde die Rechnung zum Überprüfen einen anderen Hash erzeugen, welcher nicht mehr mit $\omega + \delta$ übereinstimmt. Insgesamt ist das partiell blinde Signaturen Schema etwas umständlicher als Chaums blinde Signaturen. Dennoch bietet es die Möglichkeit einen öffentlichen Wert zu kommunizieren und dessen nachträgliche Änderung zu verhindern.

3.9 Elliptische Kurven Kryptographie

Die Kryptographie über elliptische Kurven (ECC - elliptic curve cryptography) ist ein Zweig der Kryptographie welcher bereits seit 1985 besteht [Mil85] und trotz dessen nur wenig Aufmerksamkeit genießt. Hierbei geht es um das Ver- und Entschlüsseln von Nachrichten anhand von Punkten auf einem elliptischen Graph, was verglichen mit RSA, das auf der Faktorisierung großer Zahlen basiert, erhebliche Performanzsteigerung liefert.

3.9.1 Trapdoor functions

Das asymmetrische RSA Verschlüsselungsverfahren ist heute weltbekannt und wird in über 90% der Onlinekommunikation verwendet [Kee21]. Dessen Sicherheit basiert genauso wie die von ECC auf sog.

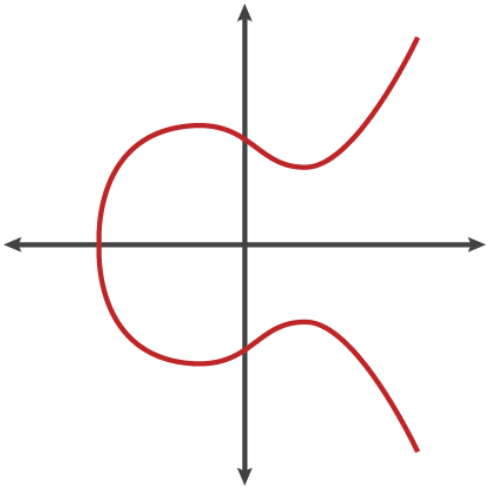

Abbildung 3.6: Mögliche Form einer elliptischen Kurve [Sul13]

Trapdoor functions. Ein Trapdoor function ist ein mathematisches Problem, welches in eine Richtung leicht zu berechnen ist, jedoch das Inverse enorm schwer. Im Falle von RSA ist die Trapdoor function das Faktorisierungsproblem. Es ist leicht 2 sehr große Zahlen miteinander zu multiplizieren. Allerdings ist es enorm schwer bei einem gegebenen Produkt dessen Faktoren zu bestimmen, insbesondere wenn die Faktoren jeweils Primzahlen sind. Dies ist der Grund, weshalb RSA sicher ist und zuverlässig die Kommunikation schützt. Bei ECC ist die Trapdoor function, die die Sicherheit der Verschlüsselung ausmacht, eine andere. Jedoch kann anhand dieser genauso ein public und private key generiert werden.

3.9.2 Funktionsweise

Um die Trapdoor function hinter ECC zu verstehen, müssen wir uns zuerst elliptische Kurven anschauen. Die algebraische Struktur von Elliptische Kurven ist eine Gruppe, welche aus einer Menge von Punkten M und einer binären Operationen $\circ$ auf 2 Punkten der Menge besteht. Die Definition einer Gruppe fordert, dass die Elemente die Eigenschaften der Assoziativität, Identität, Existenz eines inversen Elements und je nach Quelle auch Abgeschlossenheit erfüllen. [Ara16][Bog08] Zudem müssen die Koordinaten aller Punkte $(x,y) \in M$ die in der Menge liegen aus dem endlichen Feld stammen, sowie die Gleichung:

$$y^2 = x^3 + ax + b \tag{3.1}$$

erfüllen. Zusammen bildet die Menge an Punkten M einen Graph, der je nach Wahl der Parameter a,b einer Form aus Abbildung 3.6 ähnelt. Aufgrund der Gleichung 3.1 entsteht der Zusammenhang, dass ein Gerade durch 2 zufällig gewählte Punkte $P,Q \in M$, den Graph immer an genau einer dritten Stelle schneidet. Zusätzlich ist der Graph an der X-Achse durch das y^2 gespiegelt. Mit diesen Eigenschaften kann nun die Operation $\circ$ definiert werden.
Diese arbeitet wie folgt: Bei Eingabe von $A,B \in M$ finde den invertierten dritten Schnittpunkt mit dem Graph. Dieser sei hier mit C beschrieben. Eine Ausführung von $A \circ B = C$ ist in Abbildung 3.7 verdeutlicht.

Die Operation $\circ$ kann beliebig oft hintereinander mit dem jeweils neu entstehenden Punkt ausgeführt werden. Genau hier liegt die Trapdoor function von ECC. Mit Kenntnissen über der Startpunkt A und die Anzahl der Ausführungen n, ist es einfach den Endpunkt E zu berechnen. Wenn allerdings nur der Endpunkt E und Startpunkt A gegeben sind, ist es sehr rechenaufwändig die Anzahl an Ausführungen zu bestimmen.

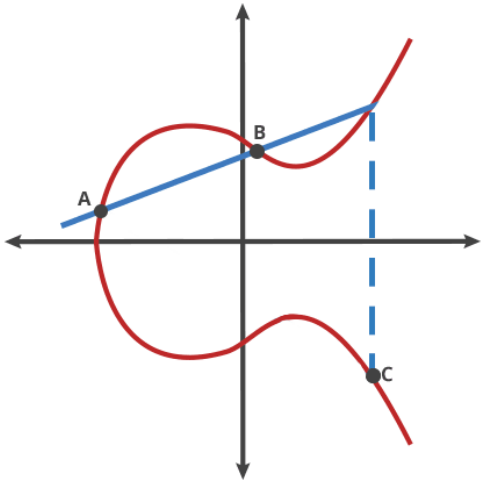

Abbildung 3.7: Operation $\circ$ auf Punkten $A, B \in M$ [Sul13]

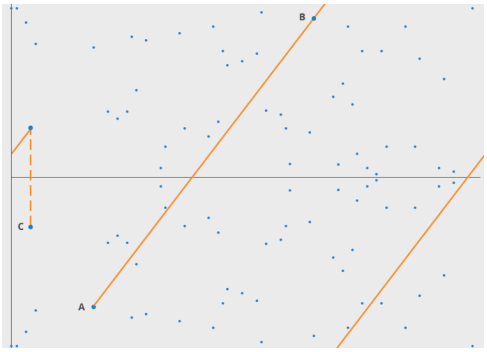

Abbildung 3.8: Graph nach Anwendung von $\mathbb{Z}_p$ [Sul13]

3.9.3 Anwendung in der Kryptographie

Allgemein ist zu beachten, dass elliptische Kurven, die für ein kryptographischen Verwendung in Frage kommen, eine andere Form haben als die eben aufgeführt Abbildungen (3.6,3.7). Entscheidend ist hierfür die Wahl des endlichen Feldes. In der Kryptographie wird hier meist $\mathbb{Z}_p$ verwendet. Diese Wahl bringt 2 Eigenschaften mit sich:

1. Das p in $\mathbb{Z}_p$ besagt, dass alle Werte $x \in \mathbb{Z}_p$, $0 \leq x \leq p-1$ erfüllen müssen. Durch die Wahl dieses Wertebereichs, können die X und Y-Achse nie den Wert p überschreiten, sondern beginnt anstatt wieder bei 0.

2. Die zweite Eigenschaft ist die ausschließliche Betrachtung ganzer Zahlen als X und Y-Koordinate. Hierdurch wird aus dem Graphen eine Menge an zufällig aussehend gewählten Punkten. Der Graph für die kryptographische Verwendung einer elliptischen Kurve ist ein Abbildung 3.8 visualisiert.

Nun muss ein passendes p sowie a, b bestimmt werden, um die genaue Menge an Punkten zu spezifizieren. Hierfür gibt es bereits eine große Auswahl und Werten in der Literatur [LM10][ML13]. Eine der prominentesten ist die "Curve25519" von Daniel J.Bernstein [Ber06]. Sie hat den Werte $p = 2^{255} - 19$ (daher der Name) und die elliptische Funktion $y^2 = x^3 + 486662x^2 + x$.

3.9.4 Vor und Nachteile

Allgemein betrachtet ist die Tatsache, dass ECC eine schneller Berechnungszeit liefert als RSA, nicht von der Hand zu weisen [Sul13]. Da die Laufzeiten von Verschlüsselung und Entschlüsselung durch RSA und ECC asymptotisch nicht gleich verhalten ist es schwer eine endgültige Antwort zu nennen. Allerdings schlägt ECC RSA bei der Gesamtzeit für Ver- und Entschlüsselung je nach Nachrichtengröße um einen Faktor von $\sim 3-20$ [MY18][Bao22]. Zudem eignet sich ECC vor allem als Ver- und Entschlüsselungsmethode auf rechenschwachen Geräten wie Mobiltelefonen aufgrund von kleineren Schlüsselgrößen[1]. So können Rechenaufwand, Energieverbrauch und RAM-Auslastung gesenkt werden [GS11].

Allerdings wurde 2007 bekannt, dass der Pseudo Random Number Generator (PRNG) Dual_EC_DRBG eine potenzielle Backdoor hat, die es Nutzern mit Informationen über einen bestimmen Wert die Zufälligkeit problemlos knacken lies [Gre13]. Dual_EC_DRBG wurde 2005-2006 zusammen von NIST (National Institute of Standarts and Technology) und der NSA veröffentlicht und basiert die Auswahl der zurück gegebenen pseudozufälligen Zahlen auf Berechnung über elliptischen Kurven. Dieser Vorfall schwächt bis heute das generelle Vertrauen in ECC. [Sul13]

3.10 AES - Advanced Encryption Standard

Der Advanced Encryption Standard ist einer der weltweit am häufigsten verwendeten symmetrischen Verschlüsselungsalgorithmen. AES wurde 2001 vom NIST als offizieller Standard für die Verschlüsselung von Daten festgelegt und hat den früher verwendeten Data Encryption Standard (DES) ersetzt, da dieser als unsicher galt. AES basiert auf dem Rijndael-Algorithmus, der von den Kryptographen Joan Daemen und Vincent Rijmen entwickelt wurde. Die hohe Sicherheit, Effizienz und Flexibilität von AES haben dazu beigetragen, dass er heute in einer Vielzahl von Anwendungen, von vertraulichen Regierungsdokumenten bis hin zur Verschlüsselung von Festplatten und Netzwerkverkehr, zum Einsatz kommt.

AES ist ein blockbasierter Verschlüsselungsalgorithmus, der Daten in Blöcken von 128 Bit verarbeitet. Dabei verwendet AES symmetrische Schlüssel, was bedeutet, dass derselbe Schlüssel sowohl für

1. Um eine security bit level von 256bits mit RSA zu erreichen, ist eine Schlüssellänge von 15360bits nötig während es bei ECC lediglich 512bits sind.[MY18]

die Verschlüsselung als auch für die Entschlüsselung der Daten verwendet wird. Der Algorithmus unterstützt Schlüsselgrößen von 128, 192 und 256 Bit, wobei größere Schlüssel im Allgemeinen eine höhere Sicherheit bieten, jedoch auch höhere Rechenressourcen erfordern. AES durchläuft mehrere Runden von Transformationen, die die Klartextdaten in scheinbar zufällige Chiffretextdaten umwandeln. Bei der 128-Bit-Schlüsselgröße umfasst der Algorithmus 10 Runden, bei 192-Bit sind es 12 Runden, und bei 256-Bit werden 14 Runden verwendet. Jede dieser Runden besteht aus mehreren Schritten, darunter Substitution (Byte Substitution mittels S-Box), Zeilenverschiebung (ShiftRows), Spaltenmischung (MixColumns) und die Hinzufügung eines Rundenschlüssels (AddRoundKey).

Das zentrale Sicherheitsmerkmal von AES ist seine Widerstandsfähigkeit gegen bekannte Kryptoangriffe wie Differenzial- und Linearkryptanalyse. Dies wird durch eine starke Schlüsselabhängigkeit und die Nutzung mehrerer kryptographischer Prinzipien, wie der Substitutions-Permutations-Netzwerkstruktur (SPN), erreicht. Darüber hinaus spielt die sogenannte Avalanche-Eigenschaft eine entscheidende Rolle: Eine kleine Änderung im Klartext oder im Schlüssel führt zu drastischen Änderungen im Chiffretext, wodurch es Angreifern erschwert wird, Rückschlüsse auf den ursprünglichen Klartext oder den verwendeten Schlüssel zu ziehen.

Verglichen mit asymmetrischen Verschlüsselungsverfahren fehlen AES jedoch gewisse Schutzziele die über die letzten Jahre zunehmend an Wichtigkeit gewonnen haben. Die Verwendung von einem einzelnen Schlüssel führt dazu, dass es die Integrität einer Nachricht nicht gezeigt werden kann, was bedeutet, dass Veränderungen an der Nachricht, die nicht vom ursprünglichen Sender stammen nicht festgestellt werden können. Aus diesem Grund haben Sadi Arman, Shamim Al Mamun und Nuray Jannat einen modified AES Algorithmus (MAES) veröffentlicht, der durch die Verwendung weitere Bit Verschiebeoperationen eine Senderauthentifizierung, Empfängerauthentifizierung und Nachrichtenintegrität gewährleisten kann [AAJ24]. Der Algorithmus beschränkt die für AES üblichen Schlüssellängenauswahl von 128, 192 und 256-bit Schlüsseln auf ausschließlich 128-bit Schlüssel, erhöht dafür aber gleichzeitig die Datenübertragungsrate um bis zu 20% [AAJ24].

3.11 Abschluss

Aus dieser Liste an verwandten Arbeiten geht hervor, dass es noch kein Bezahlsystem gibt welche den in 3.1 aufgestellten Anforderungen gerecht wird. Die Tabelle 3.1 gibt dabei einen Überblick über die vorgeschlagenen Lösungsansätze und welche der Anforderungen durch sie nicht erreicht werden.

	GNU Taler	Bitcoin	Ethereum	Privacy Pass
Anonymität	×	~	~	✓
Zeitsensitivität	✓	×	×	✓
Skalierbarkeit	✓	~	✓	✓
Vertraulichkeit	✓	✓	✓	~
Integrität	✓	✓	✓	~

Tabelle 3.1: Erfüllung nichtfunktionaler Anforderungen

GNU Taler erfüllt jede der Anforderungen abgesehen von der Anonymität. Zwar schützt GNU Taler die Anonymität des Käufers und hält dessen Privatsphäre aufrecht, allerdings sollte ein im Datentreuhändermodell verwendetes Bezahlsystem hauptsächlich die Privatsphäre des Verkäufers schützt, welche von GNU Taler bewusst öffentlich gemacht wird.

Bitcoin und Ethereum teilen sich aufgrund der Verwendung einer Blockchain das Problem von nicht zeitsensitiven Transaktionsdauern. Mit einer geschätzten Wartezeit von einer Stunde bzw. 6 Minuten, bis eine Transaktion mit ausreichend Sicherheit bestätigt werden kann, ist die Verwendung der beiden Technologien unter Einhaltung der Anforderungen nicht möglich. Des Weiteren kann argumentiert wer-

den, dass durch den public ledger der alle Transaktionen pseudonymisiert veröffentlicht, die Anonymität nur in Teilen gewahrt werden kann.

Vertraulichkeit und Integrität werden durch Privacy Pass nicht native unterstützt. Da das Protokoll ausschließlich für das Umgehen von CAPTCHAs bei CDNs entwickelt wurde, ist von einer https Verbindung auszugehen, welche diese Schutzziele gewährleistet. Jedoch sieht Privacy Pass selbst keine Maßnahmen zum Schutz der Kommunikation vor.

Aufgrund dessen, dass keines der zuvor genannten Bezahlsysteme sämtliche Anforderungen erfüllt, muss für den Anwendungsfall im Datentreuhändermodell eines neues Bezahlsystem entworfen werden. Dieses Bezahlsystem wird in dem folgenden Kapitel genauer beschrieben und spezifiziert.

4 Entwurf der Systeme

4.1 Ziel

Um einen privatsphäreschützenden Anreiz für die Benutzung eines Datentreuhänders zu schaffen, wird im Folgenden ein Bezahlsystem vorgestellt. Dieses Bezahlsystem erlaubt es Datengebenden für ihre Daten im Gegenzug einen finanziellen Ausgleich zu verlangen, ohne dabei persönliche Informationen preisgeben zu müssen. Weder der Datennutzende noch der Datentreuhänder kann bei einer regulären Transaktion bestimmen, welcher Datengebende eine Bezahlung erhält. Der hierfür verwendete Ablauf sieht vor, dass ein Datengebender zuerst seine Daten an den Datennutzenden überträgt und ihm etwas Zeit zur Verwertung der Daten lässt. Ein Datennutzender erhält maximal 12 Stunden Zeit, um den Zahlvorgang abzuschließen. Schafft er dies nicht innerhalb der 12 Stunden, so kann der Datentreuhänder ihn von der weiteren Verwendung der Plattform ausschließen.

Damit der Datengebende seine Anonymität nicht ausnutzen kann, gibt es 2 Mechanismen zur Betrugserkennung. Der erste ist eine Offenlegung der Kommunikation im Streitfall. Sollte sich eine der beiden Parteien betrogen fühlen, so kann sie vom Datentreuhänder eine Schlichtung verlangen. Dafür werden die für die Kommunikation verwendeten Einmalschlüssel dem Datentreuhänder mitgeteilt, sodass dieser alle Schritte nachvollziehen kann und entscheiden kann, ob ein Betrug vorliegt oder nicht.

Der zweite Mechanismus ist eine Reputationsvergabe. Wenn ein Datengebender besonders qualitativ gute Daten liefert, wird er durch den Datennutzenden mehr bezahlt, was sich in der Reputation widerspiegelt. Wenn er hingegen besonders schlechte Daten liefert, so kann ein Datennutzender sich weigern ihn zu bezahlen, was die Reputation in herunterzieht. So kann bei jeder Nachfrage nach Daten von einem Datennutzenden ein Reputationsschwellwert festgelegt werden, um Datengebende mit zu schlechter Reputation vor dem Einreichen ihrer Daten abzuhalten.

4.2 Geld als Anreiz

Da im Datentreuhandmodell ein Datennutzender meist durch ein Unternehmen oder eine Forschungseinrichtung repräsentiert wird, ist es verstellbar, dass der Datennutzende selbst eine Dienstleistung anbieten oder in Aussicht stellen kann. Hier wurde aber Geld als Anreiz gewählt, da die Verwendung solcher Dienstleistungen als Anreiz einige Nachteile mit sich ziehen kann.

1. **Begehrtheit** Kein Handelsgut oder Dienstleistung ist so universelles begehrt wie Geld. Zwar ist es möglich das ein Business welches hier als Datennutzender auftritt, für die Preisgabe der Daten eine Dienstleistung anbietet. Bspw. können Social Media Plattformen wie X kostenlose zeitlich bedingte Premiumabonenments oder vergleichbar Angebote in Aussicht stellen. Jedoch sind ist das Interesse an solchen Angeboten subjektiv, was sie als mögliche Anreize zwar denkbar aber eher suboptimal macht.

2. **Hoher Aufwand** Wenn der Anreiz durch den Datennutzende gestellt wird, wird dieser in jedem einzelnen Fall unterschiedlich sein. Aus der Kommunikationsstruktur folgt das alle unterschiedlichen Kompensationen auf irgendeinem Weg über der Datentreuhänder laufen müssen. Daraus ergibt sich ein enormer zusätzlicher Aufwand an Entwicklung und Implementation für jeden Datennutzenden.

3. **Verfügbarkeit** Manche Datennutzende wie bspw. Forschungsinstitutionen haben keine Dienstleistungen oder Produkte, die sie Datengebenden anbieten können. Sie sind ausschließlich an der

Forschung interessiert und können den Datengebenden nichts bieten, was sie zum Preisgeben ihrer Daten veranlassen würde.

4. **Wahrung der Anonymität** Viele C2B Treuhänder legen Wert auf die Anonymität oder Pseudonymität der Datengeber. Diese kann jedoch dadurch, dass eine Datennutzender die versprochene Dienstleistung kontrolliert, verloren gehen. Es ist bspw. denkbar, dass ein Datennutzender einen vielversprechenden Anreiz verspricht, diesen aber hinter einer Registrierung und Angaben von persönlichen Daten verbirgt, so dass die durch den Treuhänder etablierte Anonymität umgangen wird.

5. **Einhalt der Leistung** Der Datentreuhänder kann bei der Auslagerung von Anreizen an den Datennutzenden nicht zuverlässig kontrollieren, dass der Datennutzende die versprochene Dienstleistung wirklich einhält.

Durch eine die Verwendung von Geld als Anreiz werden diese Punkte umgangen. Das Interesse der Datengebenden ist nun ausschließlich abhängig von dem Ruf und dem persönlichen Interesse an einem Datengebenden und nicht von dessen Anreizangebot. Der Anspruch auf den versprochenen Anreizes ist für jeden Datennutzenden gleich implementiert und hängt maximal von der Wahl anderer Parameter ab. Es kann davon ausgegangen werden, dass alle Datennutzenden über Geld verfügen und dieses als Kompensation verwenden können. Aufgrund des Stellenwertes von Geld in unserer Gesellschaft, sind bereits viele Wegen entwickelt wurden, wie dessen Wert anonym oder pseudonym an andere überträgt. Der Datentreuhänder hat die volle Kontrolle, dass ein zu zahlender Geldbetrag vom Datennutzenden abgegeben wird und auch bei Datengebenden ankommt.

4.3 Annahmen über das Datentreuhändermodell

Datentreuhänder können in verschiedenen Variation eingesetzt werden. Abschnitt 2.2 gibt darüber bereits einen Überblick. Hier werden im Annahmen getroffen, die das Datentreuhändmodell betrefen, für welches das Anreizsystem entworden wird. Dadurch soll verdeutlicht werden, in was für einem Datentreuhändermodell das System am besten eingesetzt werden kann.

Zuerst sei anzunehmen, dass es sich bei dem Datentreuhänder um einen freiwilligen Customer to Business Treuhänder handelt. Dies ist dadurch begründet, dass bei verpflichtenden Datentreuhändern die Benutzung vorgeschrieben ist und ein Anreiz zur Benutzung für solche ein System die Nutzerzahlen nicht beeinflusst. Weiter haben bei Business to Business Treuhändern beide Parteien einen Profit aus dem Tausch der Daten. Beide sind gleichzeitig am Daten empfangen und am Daten versenden. Hier fällt daher eine deutliche Einteilung in Datengebenden, die Anreize erhalten und Datennutzende die Anreize bieten schwer. Zusätzlich ist die Aussicht auf Datengewinn durch einen Tausch, häufig Anreiz genug um ein Unternehmen zur Verwendung eines B2B Datentreuhänders zu bewegen. proof

Die beim Datentreuhänder abgelegten Daten sind verschlüsselt. Der Datentreuhänder speichert lediglich eine lange Liste an Bytes, aber kann anhand der verschlüsselten Bytes nicht feststellen, was die Daten beinhalten, welche Daten von welchem Datengebenden stammen oder wo ein Datensatz aufhört und der nächste beginnt. Er weiß also nichts über die bei ihm abgelegten Daten. Damit ein Datengebender trotzdem dem Zugang zu seinen Daten behält, wird ihm beim hochladen seiner Daten Werte für den kontreten Ablageort seiner Daten, sowie einen passenden Schlüssel zur Entschlüsselung mitgeteilt. Wenn er seine abgelegten Daten nun einsehen möchte, so kann er den Satz an Bytes von dem Speicherort anfragen und den erhaltenen Geheimtext mit Hilfe des Schlüssels entschlüsseln, was seine ursprünglich hochgeladenen Daten zum Vorschein bringt.

Es gibt 2 Paradigmen bezüglich der Auswahl von Datengebenden. Das erste davon umfasst, dass ein Datengebender dem Datentreuhänder erklärt, dass er seine Daten zum Verkauf anbieten möchte. Der Datentreuhänder bewirbt darauf die Daten des Datengebenden auf einer zu einem Marktplatz vergleichbaren Plattform und Datennutzende wählen sich die Datengebende ihre Wahl aus und beziehen von ihnen die angeboten Daten. Das zweite Paradigma dreht die Zuständigkeit um. Hier kann ein

Datennutzender einen Aufruf starten, der Datengebende dazu aufruft, Daten eines bestimmen Typen zu verkaufen. Jetzt können sich Datengebende durch eine Liste an Aufrufen durcharbeiten und den richtig für sich heraussuchen. Sobald ein passender Aufruf entdeckt wurde, kann ein Datensatz des vom Datennutzende angefragen Datentyps vorgelegt werden. Im Folgenden wird vom zweiten Paradigma ausgegangen, da es sowohl Rechenaufwand- als auch Privatsphärenvorteile liefert. Das Erstellen eines Aufrufs, sowie Weiterleiten zum Datentreuhänder, sollte beides in $O(1)$ geschehen. Natürlich hängt die Laufzeit von der konkreten implementierung ab, allerdings beinhaltet das Erstellen eines Aufrufs nur die Zusammensammlung von Informationen wie Datentypen, Namen oder Beschreibung. Es sind in meisten Fällen keine iterativen Vorgänge aktiv, die die Laufzeit in $O(n)$ bringen könnte. Im Anschluss sind keine weiteren Berechnungen von Nöten um Daten zu empfangen. Bei der individuellen Auswahl von Datengebenden und der Nachfrage nach deren Daten entsteht hingegen ein Zusammenhang zwischen der Anzahl an Datensätzen und des Rechenaufwands des Datennutzenden. Da der Datennutzende für jeden Datengebenden eine Bewertung erzeugen muss, die entscheidet ob die Daten erworben werden oder nicht, ergibt sich für den Erhalt von n Datensätzen ein Rechenaufwand von $O(n)$. Die Minimierung des Rechenaufwands für den Datennutzende ist vor allem interessant, da Datennutzende durch Unternehmen repräsentiert werden, welche in Extremfällen mehrere tausend Anfragen pro Sekunden bearbeiten müssen, während Datengebende an ihren Endgerät einen deutlich überschaubareren Rechenaufwand leisten müssen. Privatsphärenvorteile werden durch das zweite Paradigma ebenfalls erbracht, da die Einreichung von Daten so vollständig Anonym ablaufen kann. Solange durch die Kommunikation zur Mitteilung des Speicherorts und Datenschlüssels keine Informationen preisgegeben werden, kann der Datennutzende den Datengebenden unmöglich deanonymisieren. Die von ihm erhaltenen Datenparameter können dazu benutzt werden Daten beim Datentreuhänder zuerfragen. Dieser weiß genauso wenig wessen Daten angefragt wurden. Im ersten Paradigma hingegen müssen mindestens die Informationen die ein Datennutzender braucht, um eine aussagekräftige Einschätzung über die Daten trefen zu können bereitgestellt werden. Je nach dem welche Informationen davon betroffen sind besteht eine Möglichkeit die Angebote eines Datengebenden miteinander zu verlinken und so Informationen über ihn zu gewinnen.

4.4 Generelles

Im Gegensatz zu GNU Taler soll das Bezahlsystem hauptsächlich den Datengebenden und dessen Informationen schützen. Die Privatsphäre des Datennutzenden ist zwar ebenfalls schützendswert aber eher von geringerer Bedeutung, da im Datentreuhändermodell der Datennutzende meist durch ein Unternehmen oder eine Forschungseinrichtung dargestellt wird. Hinzu kommt, die Struktur eines Datentreuhänders, die den Datennutzende dazu veranlasst, eine Menge an Informationen anzugeben, um vom Datentreuhänder zugelassen zu werden. Der Datengebende hingegen ist eine einzelne Privatperson, die ihre gesammelten Daten bewusst an Unternehmen oder Forschungseinrichtungen verkaufen möchte. Der Schutz der Privatperson ist hier daher von größerem Interesse.

Unter diesen Umständen ist eine herkömmliche Überweisung per Bank deutlich zu unsicher. Sie benötigt Information wie die IBAN des belasteten Kontos und die IBAN des empfangenden Kontos, um zu wissen, von wo nach wo eine Geldsumme fließen soll. Dadurch werden bereits zu viele Informationen über den Datengebenden bekannt, da der Datennutzende nun genau heraus finden kann, von wem die erworbenen Daten stammen und ob er bereits vorher schon von demselben Datengebenden Daten gekauft hat. Zusätzlich kann die Bank nachvollziehen, dass eine Privatperson sich als Datengebender angemeldet hat und wem sie ihre Daten bereits verkauft hat. Um dieses Problem zu lösen, wird hier eine der gängigsten anonymen Onlinezahlmethoden verwendet: Kryptowährungen. Die Abschnitte 3.2, 3.3, 3.4, 3.5 führen bereits wenige davon auf und beschreiben, warum sie nicht im Datentreuhändermodell verwendbar sind. Aufgrund dessen wird hier eine neuer Coin eingeführt. Jedoch ist dieser weder Blockchain basiert noch für Spekulationszwecke geeignet. Ein Coin kann - so wie bei vielen Kryptowährungen üblich - ausschließlich bei einem Exchange gegen Geld getauscht werden. Bei dem

initialen Tauschen mehrerer Coins ist der monetäre Wert in Euro fest im Coin gespeichert und kann nicht verändert werden.

An dieser Stelle ist anzumerken, dass keine Regulierungen oder Identifizierung zum Erwerben eines Coins benötigt werden. Jedoch sind Datennutzende die einzigen die Coins sicher ausgegeben können. Das wird dadurch deutlich, dass das Bezahlsystem nur dazu verwendet werden kann, Coins von Datennutzenden zu Datengebenden zu übertragen. Datengebende können zwar selbst auch Coins erwerben, allerdings ist der Handel nicht über das hier eingeführt Bezahlsystem möglich. Daher bleibt nur der reguläre Tausch, welcher außerhalb des Systems stattfindet und keine Betrugserkennung besitzt. Unter der Berücksichtigung dass das Einzige, was dem Coin einen festen Wert sicherstellt, die Signatur des Exchanges ist, kann leicht ein nicht verifizierter Coin erstellt werden und im unsicheren Tausch angeboten werden. Der andere Tauschpartner hat keine Möglichkeit den Coin vor Abwicklung des Tauschs auf seine Echtheit zu prüfen, was ein riesen Potentiall für Betrug bietet. Aufgrund dessen ist anzunehmen, dass nur Datennutzende Coin erwerben werden, da der Tausch unter Datengebenden deutlich zu unsicher ist.

Nachdem ein Datennutzender einen Coin erworben hat, kann er diesen lokal speichern, bis er ihn ausgeben möchte. Sobald ihm ein Datengebender seine Daten mitteilt, hat der Datennutzende 12 Stunden Zeit die Daten auszuwerten. Wenn ein Datennutzender nach Ablauf der 12 Stunden noch keine Transaktion zu dem Datengebenden abgeschlossen hat und dieser bei dem Datentreuhänder eine Betrugsanklage gegen den Datennutzenden vorlegt, so kann der Datentreuhänder sich dazu entschließen den Datennutzenden entweder zeitlich bedingt oder dauerhaft von der weiteren Verwendung der Plattform auszuschließen. Ein gutwilliger Datennutzender kann nach der Auswertung der Daten eine passende Summe an Coins aus seinem lokalen Speicher auslesen und an den Datengebenden sendet. Sobald dieser die Coins empfängt, prüft er die Signatur der Coins. Wenn die Signatur mit der des Exchanges übereinstimmt, kann er sie an den Exchange weiterleiten, um sich den monetären Wert bspw. auf sein Konto gut schreiben zu lassen. Stimmt die Signatur nicht mit der des Exchanges überein, so kann der Datengebende auch hier wieder einen Betrug beim Datentreuhänder melden.

4.5 Verschlüsselung

Da in der Kommunikation indirekt Geld transferiert wird, ist es von höchster Priorität, die gesendeten Nachrichten vor dem Mithören oder Verändern durch unbefugte dritte zu schützen. Für diesen Schutz wird eine Mischung aus der in 3.9 erklärten elliptischen Kurven Kryptographie und dem in 3.10 angeführten MAES-Algorithmus verwendet.

Mit der Benutzung eines public Keys bei ECC wird sichergestellt, dass nur Personen mit Wissen über den zugehörigen private Key die Nachricht lesen können. Gleichzeitig wird ein Hash der unverschlüsselten Nachricht gebildet und mit dem private Key des Senders signiert. Auf diesem Weg kann der Empfänger die Nachricht entschlüsseln, selbst den Hash bilden und prüfen, ob der signierte Hash mit dem selbst gebildeten übereinstimmt. Dadurch kann sowohl die Vertraulichkeit als auch die Integrität der Nachricht sichergestellt werden.

AES hingegen verwendet nur einen Schlüssel, der gleichzeitig zum Verschlüsseln und Entschlüsseln einer Nachricht gebraucht wird. Es erreicht erheblich schnellere Berechnungszeiten als ECC, da AES auf Operationen basiert, welche lediglich bit verschieben, aber keine komplexen mathematischen Berechnungen durchführen. Da AES aber keine Integrität gewährleisten kann, wird hier für die symmetrische Kommunikationsverschlüsselung der MAES-Algorithmus verwendet. Durch ihn kann wie bei ECC eine unbemerkte Veränderung der Nachricht ausgeschlossen werden, während trotzdem nur ein Schlüssel zum Ver- und entschlüsseln benötigt wird.

Es sei angemerkt, dass von hieran folgend jede Nachrichtenübertragung zwischen 2 Akteuren, dies mögen bspw. Datentreuhänder und Datengebender sein, immer via ECC verschlüsselt und signiert ist, es sei denn es wird für den Kommunikationsschritt anders definiert.

4.6 Coin Generierungsphase

Hier werden die Coins die im späteren Verlauf beider Systeme verwendet werden vom Datennutzenden erstellt und vom Datentreuhänder signiert. Die einzelnen Schritte sind in Abbildung 4.1 verdeutlich und werden im Folgenden beschrieben.

4.6.1 Ablauf

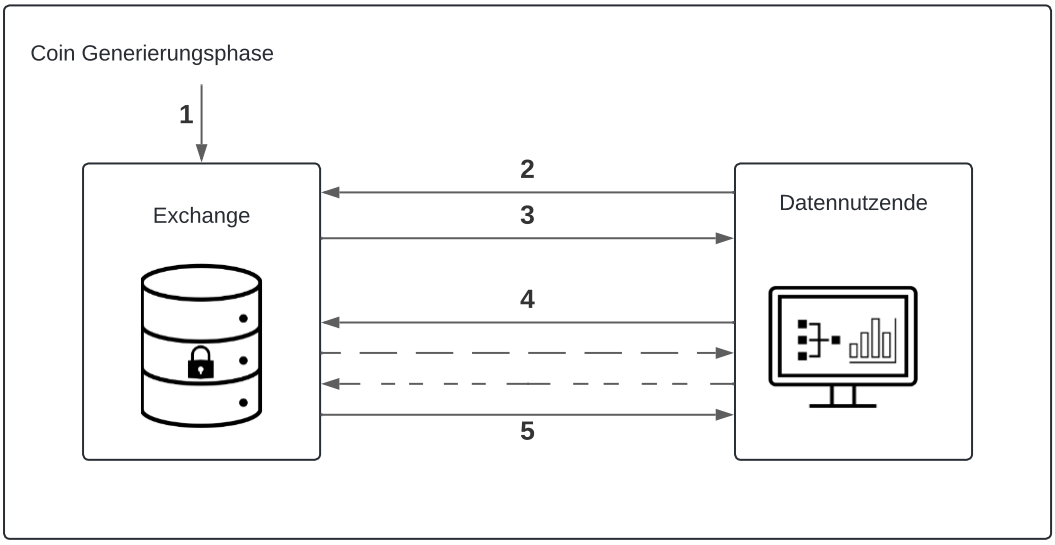

Abbildung 4.1: Coin Generierungsphase Ablauf

1. Zahlungseingang $(apk \leftarrow AccountPublicKey, ES \leftarrow ErhalteneSumme)$
Der Exchange erhält über einen beliebigen Weg eine Summe und einen Konto public Key. Die genaue Umsetzung des Zahlungseingang liegt bei dem Exchange. Er kann alles zwischen einer Banküberweisung mit der apk als Verwendungszweck, bis hin zu einem Briefumschlag mit Bargeld und einem leserlich aufgeschriebenen apk sein. Nach Eingang einer Zahlung erstellt der Exchange einen sogenannten $CoinGenerationToken$ bestehend aus $(nonce, apk, ES, spent \leftarrow false)$, verschlüsselt ihn mit dem apk und speichert diesen intern.

2. CoinGenerationToken abrufen (apk)
Da der Exchange nur den apk erhält, hat er keine Möglichkeit den $CoinGenerationToken$ einem Datennutzenden zuzuweisen. Jetzt kann jeder beliebige Nutzer alle Einträge für einen apk abfragen. Allerdings ist nur der Datennutzende in der Lage den $CoinGenerationToken$ zu entschlüsseln, der den passenden secret Key zum apk hat. Dies ist im Idealfall nur der Datennutzende der die Transaktion getätigt hat.

3. CoinGenerationToken übertragen $([\{(nonce, apk, ES)\}_{apk}])$
Auf eine $CoinGenerationToken$ Anfrage des Datennutzenden antwortet der Exchange mit allen noch nicht eingelösten Token für den apk. Nachdem der Datennutzende die Token erhalten hat, kann er diesen zuerst mit dem passenden secret Key entschlüsseln. Im Anschluss kann er selbst Coins erstellen. Ein Coin besteht aus $(nonce, value)$. Bei der Erstellung sind 2 Sachen zu beachten. Zuerst muss der summierte Wert aller erstellten Coins gleichgroß sein wie die ES des Tokens. Des Weiteren gibt es eine Menge an möglichen Werten PV, sodass $\forall value \in PV$ gilt. Dies ist vor allem wichtig, da $value$ beim Signieren und Einlösen des Coins für den Exchange sichtbar ist und dieser bei einer Wahl eines selten vorkommenden $value$ eine Verbindung zwischen den Phasen herstellen kann.

4. Signierung Anfragen $((nonce, apk, ES), [value])$

Eine Signieranfrage ist der Start eines Durchlaufs von partiell blinden Signaturen, bestehend aus einem $CoinGenerationToken$ und einer Menge an $values$. Zuerst kann der Exchange prüfen, ob der gesendete Token noch nicht eingelöst wurde. Anschließend summiert er alle $value$ auf und prüft, ob die Summe gleich dem ES des Tokens ist. Wenn beide Überprüfungen akzeptieren, kann der Exchange mit dem partiell blind signieren anfangen und bei seinem gespeichert Token $spent \leftarrow true$ setzen, um eine doppelte Einlösung zu verhindern.

Gestrichelte Pfeile $([a, b]), ([e])$

Die Pfeile zwischen 4. und 5. sind hier gestrichelt eingetragen, da sie notwendige Kommunikationsschritte von partiell blinden Signaturen abbilden. Sie sind essenziell für die Funktionsweise und wurden bereits in 3.8 erklärt, weshalb sie hier nur zur Vollständigkeit aufgelistet werden.

5. Signieranfrage Antwort $([r, c, s, d], \pi_{sk})$

Vorausgesetzt alle Überprüfungen aus Schritt 4 akzeptieren, so erhält der Datennutzende nun eine Menge an partiell blinden signierten Coins und kann diese für spätere Verwendung lokal speichern. Zusätzlich berechnet der Exchange einen BDLEQ auf dem gleichen Weg, wie er in 3.5 beschrieben ist. Nachdem der Datennutzenden diesen verfiziert hat, kann er sicher sein, dass der Exchange zum signieren seiner Coins den selben private Key verwendet hat wie für die Signatur aller Coin Generierungsphasen. Sollten die Überprüfungen nicht akzeptieren, so kann der Datennutzende entweder bei Schritt 4 mit anderen Coins oder bei Schritt 2 wieder ansetzen.

Auf diese Weise kann ein Datennutzender Geld bei einem Exchange einzahlen und eine dem Geldbetrag entsprechende Menge an Coins erhalten, ohne dass der Exchange die von ihm ausgehändigten Coins nachverfolgen kann. Gleichzeitig ist es für einen Datennutzende nicht möglich Coins zu erhalten für die er keinen monetären Gegenwert bereitgestellt hat.

4.6.2 Finanzierung des Exchange

Zusätzlich ist zu erwähnen, dass Exchanges finanziell motiviert sein können oder eine Gebühr verlangen möchten, um bspw. laufende Serverkosten zu decken. Sollte dies der Fall sein, kann vor dem Beginn der Coin Generationphase vom dem Exchange eine feste Gebühr ausgerufen werden. Wenn sich ein Datennutzender trotz der Gebühr dazu entscheidet diesen Exchange zu verwenden, so kann der Exchange nach dem Eingang einer Zahlung in Schritt 1, einen $CoinGenerationToken$ erstellen, welcher die Gebühr bereits abzieht. Wenn also ein Datennutzender eine Transaktion von 100€bei Exchange mit 10% Gebühr tätigt, so erstellt der Exchange einen $CoinGenerationToken$ für den angegebenen apk mit $ES \leftarrow 90$€. Auf diese Weise kann eine Exchange seine finanziellen Interessen geltend machen. Auch wenn es nicht das Ziel eines Exchanges sein sollte wirtschaftlich zu handeln, lässt so immerhin verhindern, dass ein Exchange mit den Betriebskosten Minus macht.

4.7 Bezahlvorgang

Nun wo die Coins erstellt und signiert sind, kann damit angefangen werden sich mit dem Ausgeben dieser Coins zu beschäftigen. Dafür gibt es das Bezahlsystem welches es Datennutzenden erlaubt die Daten von Datengebenden anzufragen. Diese können der Anfrage nachkommen, wofür sie der Datennutzende im Gegenzug mit den Coins entlohnt. Damit bei dem Vorgang die Privatsphäre des Datengebende so gut wie möglich geschützt wird, findet der Austausch der Daten und Coins über Postfächer beim Datentreuhänder statt. Dadurch entsteht keine direkte Kommunikation zwischen Datengebenden und Datennutzenden welche potenziell schützenswerte Informationen wie IP oder dergleichen preisgibt. Wie der Ablauf genau funktioniert wird im Folgenden erklärt.

4.7.1 Ablauf

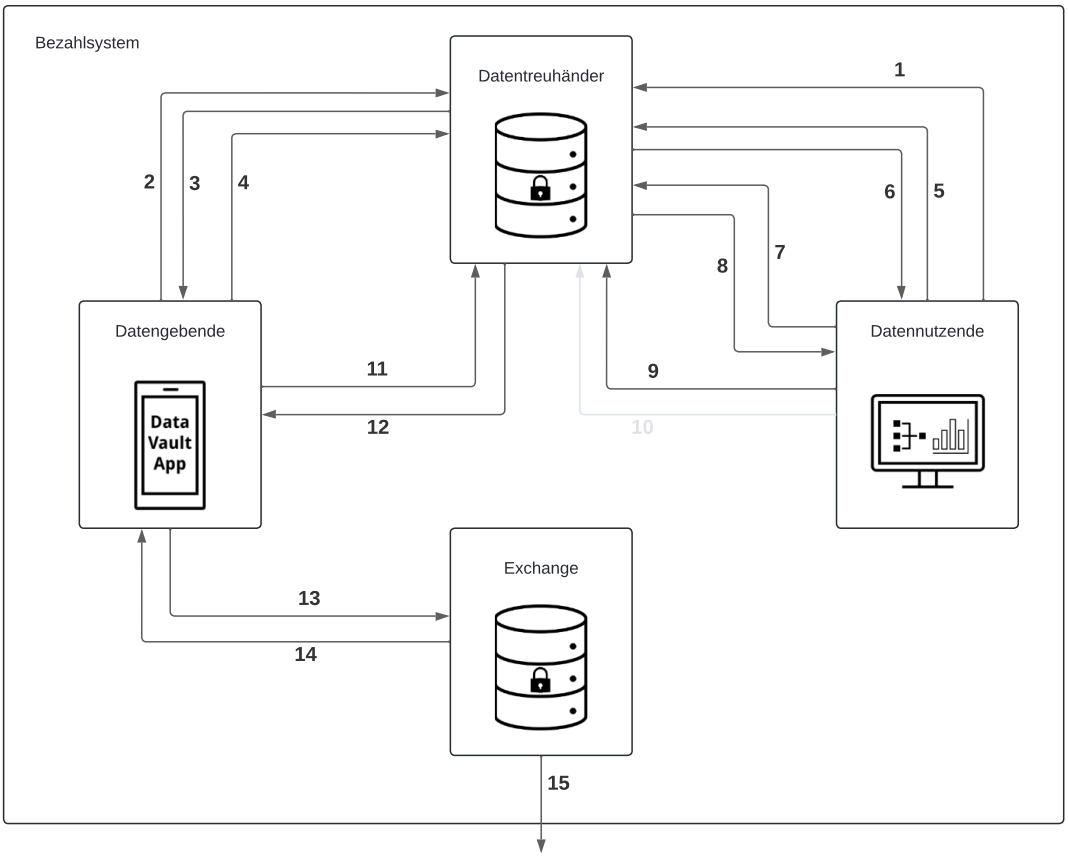

Abbildung 4.2: Bezahlsystem Ablauf

1. Aufruf starten $(datatype, reputationThreshold, price)$
Der Datennutzende erstellt beim Datentreuhänder einen Aufruf zur Datenteilung. Damit verkündet er an alle Datengebenden, dass er Daten von einem bestimmten Typ sucht und nennt direkt wieviel er für diese Daten bereits ist zu bezahlen. Da der Datennutzende nach Anlegung des Aufrufs direkt Daten erhalten kann, für die er Zahlen muss, braucht es eine Möglichkeit ein gewisses Vertrauensminimum festzulegen. Der $reputationThreshold$ gibt dabei einen Schwellwert an, so dass der Aufruf nur für Datengebende mit einer Reputation über diesem Schwellwert sichtbar ist.

2. Aufrufliste anfragen $(DG\text{-}ID)$

Erst nach reputationssytem

3. Aufrufliste Antwort $([callDetails, DN\text{-}publicKey])$
Nachdem der Datengebenden seine Reputation bewiesen hat, kann der Datentreuhänder seine lokale Datenbank mit Aufrufen durchsuchen und alle Aufrufe welche einen Reputationsschwellwert niedriger als die bewiesen Reputation haben in eine Liste aufnehmen. Damit besteht jeder Eintrag der List dem public Key des Datennutzenden, der den Aufruf gestartet hat und Aufrufdetails die wichtige Informationen bezüglich des Aufrufs geben. Die genauen Inhalte der Aufrufdetails sind Treuhänderabhängig, da je nach Gebiet des Treuhänders verschieden Details über den Datennutzende für die Entscheidung des Datengebenden wichtig sein können. In den meisten Fällen umfassen die Aufrufdetails Daten wie die angefragte Datentypkategorie (GPS-Daten, persönliche Daten, etc.), den Namen des Datennutzenden, den Preis den jeder Datengebende ausgezahlt bekommen soll, einen Titel oder eine Beschreibung. Auch Informationen darüber, ob es sich bei dem Datennutzenden um eine Forschungseinrichtung handelt, werden können hier kommuniziert werden.

4. Daten teilen $(DN\text{-}pk, \{sharedKey\}_{DN-pk}, \{referenceCode, dataLocation, dataKey\}_{sharedKey})$
Der Datengebende hat die genannten Aufrufdetails erhalten und sich dazu entschieden, seine Daten für einen Aufruf mit einem Datennutzenden zu teilen. Generell gilt wie in 2.2 erklärt, dass ein Datengebender seine Daten nicht selbst speichert, sondern sie verschlüsselt beim Datentreuhänder lagert. Er behält nur den Speicherort und Schlüssel der Daten, um jederzeit Zugriff auf diese zu haben. Um seine Daten mit einem Datennutzenden zu Teilen, genügt es daher den Speicherort und Schlüssel zu übertragen.

Für die Übertragung des Speicherorts und Schlüssels wird ein Postfach beim Datentreuhänder verwendet. So kann ein direkter Austausch zwischen Datengebenden und Datennutzenden verhindert werden. Ein Datengebender kann Ort und Schlüssel verschlüsselt in das Postfach legen und nur der Datennutzende, der das Wissen über den passenden privaten Schlüssel besitzt, kann Ort und Schlüssel wieder entschlüsseln.

Einer der Betrugserkennungsmechanismen ist die Offenlegung der Kommunikation im Streitfall, so dass der Datentreuhänder die Nachrichten einsehen kann und entscheiden kann welche der Parteien recht hat. Damit bei dieser Offenlegung nur genau die Nachrichten entschlüsselt werden, die Teil einer spezifischen Kommunikation zwischen Datengebenden und Datennutzenden sind, wird für jede Abwicklung eines Tausches ein neuer symmetrischer Schlüssel erstellt.

Damit ein Datengebender einen Eintrag in das Postfach zu Teilen von Daten senden kann, muss er zuerst die folgenden Informationen bestimmen. Zuerst wird der public Key des Datennutzenden im Klartext angegeben, da ein Datennutzender beim Abfragen des Postfaches so bestimmen kann, welche Einträge für ihn bestimmt sind. Anschließend wird ein neuer symmetrischer Schlüssel $sharedKey$ erstellt. Dieser wird mit dem public Key des Datennutzenden verschlüsselt, damit mit dieser den $sharedKey$ kennt. Nun kann der Datengebende den $sharedKey$ verwenden, um das Tupel aus $dataLocation$, $dataKey$ und $referenceCode$ zu verschlüsseln. Hierbei sind $dataLocation$ und $dataKey$ die eben beschriebenen Speicherort und Schlüssel der Daten beim Datentreuhänder. Dabei sei angemerkt, dass sowohl $dataLocation$ als auch $dataKey$ mehr als nur eine Variable sein können und von dem vom Datentreuhänder verwendeten Speicherverfahren und Verschlüsselungsverfahren abhängen. Der $referenceCode$ ist eine zufällige Zahl, die später bei Schritt 9 verwendet wird, um eine Zugehörigkeit von Übertragen der Daten und Coins herzustellen.

5. Datenpostfach abfragen $(DN\text{-}pk)$
Der Datennutzende kann sich zu einem beliebigen Zeitpunkt dazu entscheiden, alle Nachrichten die den angegeben public Key enthalten anzufragen. Dafür muss er den gewünschten public Key mitübertragen, um den Datentreuhänder zu signalisieren, an welchen Nachrichten er interesse hat. Der public Key muss hier bei jeder Anfrage spezifiziert werden, da zum einen keine Form an Authentifizierung stattfindet und jeder alle Nachrichten eines public Keys anfragen kann. Zum anderen bleibt dem Datennutzenden so die Möglichkeit offen, für jeden Aufruf einen eigenes Schlüsselpaar zu verwenden. Dies kann bspw. große Vorteile bei der internen Organisierung haben.

6. Datenpostfach Antwort $([\{sharedKey\}_{DN-pk}, \{referenceCode, dataLocation, dataKey\}_{sharedKey}])$
Nachdem der Datentreuhänder die Anfrage an das Datenpostfach erhalten hat, erstellt er eine Liste mit Postfacheinträgen, die den public Key der Anfrage im Klartext angegeben haben. Die einzelnen übertragenen Postfacheinträge können dabei auf den public Key verzichten, da dieser nur zu Identifizierung genutzt wird. Der Datennutzende kennt den public Key ohnehin da er ihn in der Anfrage definiert. Die vollständige alle Postfacheinträge kann zurück an den Datennutzenden gesendet werden.

7. Daten anfragen $(dataLocation)$
Der Datennutzende kann nun für jeden empfangen Postfacheintrag, den passenden secret Key verwenden, um den shared Key des Eintrags zu entschlüsseln. Solange der Datengebende sich an das Protokoll gehalten hat und den Schlüssel, den er zum Verschlüsseln des Datentupels verwendet hat auch als Klartextschlüssel angegeben hat, funktioniert dieser Schritt einwandfrei und der Datennutzende

erhält Zugang zu $dataLocation$, $dataKey$ und $referenceCode$. Jetzt kann er den Datentreuhänder nach den Daten an der angegebenen $dataLocation$ fragen, welche mit dem $dataKey$ verschlüsselt sind.

8. Daten Antwort $(\{data\}_{dataKey})$
Die durch die $dataLocation$ beschriebenen Daten werden durch den Datentreuhänder ausgelesen. Da auch hier wieder keine Authentifizierung stattfindet, kann jeder einen beliebigen Datensatz beim Datentreuhänder anfragen erhält die an der Stelle gespeichert Daten. Da allerdings nur der Datennutzende, welcher den $dataKey$ vorher erhalten die Daten lesen kann, ist das wahllose Anfragen von Daten beim Treuhänder sinnlos, da ohne Wissen über die $dataLocation$ nur Teile eines Datensatzes oder Überlappungen in andere Datensätze geschehen können, welche verschiedene $dataKeys$ benötigen würden. Der Datentreuhänder kann daher mit den ausgelesenen Daten antworten ohne Schlüsse ziehen zu können, wer die Daten angefragt hat oder ob die Nachfrage Teil eines Tausches zwischen Datengebenden und Datennutzenden ist.

Nach Erhalt der verschlüsselten Daten kann der Datennutzende den zu den Daten gehörenden $dataKey$ verwenden, um die Daten zu entschlüsseln und an den Klartext zu gelangen. Jetzt kann der Datennutzende die Daten für seine Zwecke verwerten.

9. Coins in Postfach legen $(referenceCode, \{[nonce, value]\}_{sharedKey})$
Mit der Verwertung abgeschlossen kann der Datennutzende nun den Datengebenden bezahlen. Dafür lädt er eine Liste an Coins aus seinem Speicher die in der Summe gleich dem im Aufruf ausgeschrieben Preis sind. Er verwendet den in Schritt 6 erhaltenen $sharedKey$, um die Liste an Coins zu verschlüsseln. Anschließend verwendet er den ebenfalls in Schritt 6 erhaltenen $referenceCode$, um zusammen einen Eintrag an das Coinpostfach zu senden.

Da der $referenceCode$ von dem Datengebenden mit dem $sharedKey$ verschlüsselt wurde, ist davon auszugehen, dass ein Eintrag mit dem gleichen $referenceCode$ nur von dem Datennutzenden stammen kann, der den Aufruf gestartet hat und die Daten erhalten und verwertet hat.

10. Coinpostfach abfragen $(referenceCode)$
Da ein Datengebender nach dem Erstellen eines Datenpostfacheintrags den dort genutzten $referenceCode$ lokal als offen abspeichert, kann er nun bei Bedarf den Datentreuhänder fragen, ob es bereits einen Eintrag in dem Coinpostfach mit diesem $referenceCode$ gibt. Er kann somit den Datentreuhänder fragen, ob er für das Teilen seiner Daten bereits bezahlt wurde.

11. Coinpostfach Antwort $(\{[nonce, value]\}_{sharedKey})$
Wenn der Datennutzende bereits einen Eintrag mit den verschlüsselten Coins eingesendet hat, so werden die unter dem $referenceCode$ angegebenen verschlüsselten Coins an den Datengebenden zurückgegeben. Der Datengebende kann daraufhin den $sharedKey$, welcher in Kombination mit dem $referenceCode$ gespeichert wurde verwenden, um die erhaltenen Coins wieder zu entschlüsseln. Wenn er anschließend die Liste an Coins erhält, kann er überprüfen, dass die Summe der Coins auch den Preis des Aufrufs ergibt und dass die partiell blinde Signatur, welche in der Coin Generierungsphase Schritt 4. und 5. ausgestellt wurde, auch tatsächlich valide ist. Sollten diese Tests beide akzeptieren, so kann der Datengebende die Coins lokal speichern, bis er sie einlösen möchte.

12. Coins einlösen $([nonce, value], address)$ Ein Datengebender kann seine angesammelten Coins jederzeit einem Exchange senden. Dafür kann er die in Schritt 11 gespeicherten Coins aus dem Speicher lesen und gesammelt oder einzeln zusammen mit einer Form an Zahlungsadresse an einen Exchange übertragen. Dabei ist die Form der $address$ vom Exchange abhängig. Bei der Implementation wird das Übertragen einer IBAN verwendet, allerdings ist es genauso gut möglich stattdessen eine Bitcoin Walletadresse oder eine beliebige andere Zahlungsmöglichkeit zu verwenden. Ein Exchange

kann auch mehrere Zahlungsoptionen gleichzeitig anbieten in welchem Fall dann die Zahlungsoption als zusätzlicher Parameter mit übertragen werden muss.

13. Bestätigung $([nonce, status \in \{true, false\}])$ Nachdem die Coins beim Exchange eingetroffen sind überprüft dieser, ob die Coins bereits vorher schon eingelöst wurden und ob die partiell blinde Signatur gültig ist. Anschließend meldet er dem Datengebenden den Status der Coins zurück. Dafür erstellt er eine Liste, die für jeden Coin den *nonce* zur Zuteilung, sowie einen boolean Wert der angibt, ob der jeweilige Coin akzeptiert wurde, und sendet diese zurück.

14. Geld austeilen $(address)$ Hier wird abschließend eine ausgehende Zahlung vom Exchange zu der angegebenen Zahlungsadresse getätigt. Die genau kommunizierten Daten sind hier von der Zahlungsoption abhängig. Allerdings muss hier sichergestellt werden, dass der ausgehende Betrag genau dem Wert der eingelösten Coins gleicht.

4.7.2 Verwendung von Proxys

Damit eine vollständige Anonymität erreicht werden kann, müssen sowohl Datengebende als Datennutzende ihre Nachrichten in den Schritten 4,9 und 10 über einen Proxy senden. Die Aufgabe des Proxys ist hier die verschleierung der IP-Adresse. Durch das Weiterleiten durch einen Proxy bleibt die wahre IP-Adresse des Datengebenden bzw. Datennutzenden geheim. Ohne einen solchen Proxy kann der Datentreuhänder über die IP-Adresse mehrere Nachrichten miteinander verbinden und über längere Zeit mitschneiden welcher Datengebende oder Datennutzende an wie viele Handeln beteiligt war. Zugegebenermaßen ist eine dauerhaft gleichbleibende IP-Adresse eine Seltenheit. Der IPv4 Adressraum ist bereits seit mehreren Jahren erschöpft, weshalb IPv4 Adresse in regelmäßigen Abständen neu an aktuell sendende Geräte verteilt werden. Trotzdem bleibt die IP-Adresse für mindestens eine Sitzung mit dem Datentreuhänder gleich, was diesem ermöglicht alle in dieser Sitzung gesendeten Nachricht zu der selben Person zurückzuverfolgen.

Die Art oder der Anbieter des Proxys ist in diesem Kontext nicht von Bedeutung. Solange er zuverlässig eingehende Nachrichten an die angegebene Ziel-IP-Adresse weiterleitet, bleibt die wahre IP-Adresse geschützt, was dem Datentreuhänder keine Möglichkeit zur Verlinkung mehrerer Nachrichten lässt.

4.7.3 Bezahlungszeitraum

Ein häufiger Betrugsversuch im Onlinehandel ist das Vortäuschen einer Zahlungsintention durch den Käufer die nicht eingehalten wird. Hier trifft dies den Fall in dem ein Datennutzender einen Aufruf startet, Datengebende in dem Vertrauen bezahlt zu werden ihre Daten weitergeben und der Datennutzende mit den gesammelten Daten verschwindet. Um Datengebende vor diesem Fall abzusichern, gibt es einen Maximalbezahlraum von 12 Stunden. Sollte also ein Datengebender innerhalb von 12 Stunden nach dem Weiterleiten seiner Daten keine Bezahlung erhalten, so ist es ihm möglich beim Datentreuhänder eine Beschwerde einzureichen. Wenn die Schuld tatsächlich beim Datennutzenden liegt und dieser keine Zahlung nachweisen kann, kann der Datentreuhänder den Datennutzenden nach wiederholtem Vorkommen dauerhaft von der weitergehenden Benutzung der Plattform ausschließen. Da zur Registrierung als Datennutzender in der Regel eine große Menge an Informationen wie Name, Branche, Verwendungszweck der Daten, etc. anzugeben sind, kann angenommen werden, dass sich ein Datennutzender nach dem Ausschluss nicht einfach unter einem anderen Namen wieder registeren kann. Außerdem wird in dem Paper ... von K. Röbert, A. See, J. Schug und M. Fischer ein Verfahren präsentiert, dass es ermöglicht Nutzer selbst nach Änderung ihres Namens eindeutig wieder zu erkennen. Dieses Verfahren kann hier eingesetzt werden um sicherzustellen, dass kein Datennutzender der bereits von der Verwendung ausgeschlossen wurde, sich erneut registeren kann.

Die Dauer von 12 Stunden kann von Treuhänder zu Treuhänder variieren. Dieser Zeitraum wurde gewählt, um einen bösartigen Datennutzende keine zu große Menge an Daten stehlen zu lassen, bevor seine Machenschaften bemerkt werden. So steigt bspw. die Menge an Daten, welcher der Datennutzende klauen kann, linear mit dem Bezahlungszeitraum, solange davon ausgegangen wird, dass der Andrang an Datengebenden, die ihre Daten teilen, zeitlich gleichverteilt ist. Gleichzeitig sollte der Zeitraum nicht zu kleine gewählt sein, da eine Nichteinhaltung schwerwiegende Folgen hat. Bei einer Wahl von wenigen Sekunden reicht z.B. ein Serverausfall des Datennutzenden, um den Bezahlzeitraum zu überschreiten, ausreichend Beschwerden zu sammeln und einen Ausschluss auszulösen.

4.7.4 Mögliche Streitfälle

Wie schon bei 4.1 erwähnt wurde, soll im Streitfall die Kommunikationsschritte, die den Austausch von Daten und die dazugehörige Bezahlung beinhalten offengelegt werden können. Wie genau diese Offenlegung funktioniert und in welchen Fällen sie benutzt wird, wird hier geschildert.

1. **Fehlerhafte Verschlüsselung.** Der erste Streitfall kann nach Schritt 6 entstehen. Zu diesem Zeitpunkt hat der Datengebende bereits die $dataLocation$ und den $dataKey$ mit dem $sharedKey$ verschlüsselt, in das Postfach des Datentreuhänders gesandt und der Datennutzende hat diesen Eintrag erhalten. Sollte nun der Datennutzende den erhaltenen Eintrag nicht entschlüsseln können, so liegt dies entweder an einem inkorrekt angewendeten, angegebenem oder übertragenen public Key oder $sharedKey$. Da der Eintrag anonym in das Postfach beim Datentreuhänder abgelegt wurde, besteht vorerst keine Möglichkeit für den Datennutzenden Maßnahmen zu ergreifen. Erst nach Ablauf der 12 Stunden, nach denen sich der Datengebende beschwert, dass er keine Coins erhalten hat, kann eine Lösung gefunden werden. Der Datengebende verweist auf den Eintrag der mit Hilfe des $sharedKey$ verschlüsslt wurde und legt den $sharedKey$ offen. Der Datentreuhänder kann probieren den Eintrag mit Hilfe des $sharedKey$ zu entschlüsseln. Sollte es ihm nicht gelingen, liegt die Schuld beim Datengebende. Wenn es ihm doch gelingt, so erhält der Datennutzende eine Chance zu zeigen, dass es den $sharedKey$ nicht lesen konnte. Dafür wird der $sharedKey$ mit dem ihm Eintrag gespeicherten public Key des Datennutzenden verschlüsselt. Sollte dabei ein unterschiedlicher Geheimtext entstehen als in dem Eintrag unter $\{sharedKey\}_{DN-pk}$ abgelegt ist, so wurde falsch verschlüsselt. Stimmen die Geheimtexte überein, so ist anzunehmen, dass der Datennutzende sowohl den Schlüssel als auch die Daten lesen kann und die Bezahlung verweigert.

   Für diese Beweisstruktur ist essenziell, dass die asymmetrische Verschlüsselung mittels ECC bei Verwendung der selben Parameter den selben Geheimtext ausgibt. Die ist nicht selbstverständlich, da für die Verschlüsselung häufig Techniken wie das Salten (hinzufügen einer Zufallsvariable) verwendet werden, um jedes Mal einen unterschiedlichen Geheimtext zu erhalten.

2. **Fehlerhafte Datenparameter.** Dieser Fall tritt nach Schritt 8 ein. Der Datennutzende hat die unter $dataLocation$ abgelegt Daten erhalten, jedoch nach entschlüsseln dieser mit Hilfe des $dataKey$ nur unleserliche Geheimtexte erhalten. Auch hier wird der $sharedKey$ und der Postfacheintrag dem Datentreuhänder mitgeteilt. Dieser kann sich durch das Entschlüsseln Zugang zur $dataLocation$ und $dataKey$ verschaffen. Daraufhin überprüft er selbst, ob er die unter $dataLocation$ abgelegten Daten mit dem $dataKey$ entschlüsseln kann. Erhält er dabei ebenfalls nur unleserliche Geheimtexte, so sind die angegebene Kombination aus $dataLocation$ und $dataKey$ inkorrekt und die Schuld liegt beim Datengebenden. Sollte er jedoch leserliche Daten vorfinden, wird dies als Betrugsversuch des Datennutzende gewertet.

3. **Nutzlose Daten.** Ähnlich wie 2 tritt dieser Streitfall nach Schritt 8 auf. Hier kann der Datennutzende die referenzierten Daten zwar lesen, empfindet diese jedoch als nutzlos. Das Vorgehen zur Offenlegung deckt sich ebenfalls mit dem von 2. Der Datennutzende gibt Postfacheintrag und

$sharedKey$ an, der Datengebende entschlüsselt den Eintrag, liest die Daten von der $dataLocation$ und entschlüsselt diese mit dem $dataKey$. Es gilt nun abzuwägen, ob die vorliegenden Daten tatsächlich nutzlos sind. Diese Bewertung ist in den meisten Fällen schwer zu entscheiden. Sollte es sich um einen völlig leeren Datensatz handeln, der nur aus Nullbits besteht, kann mit überwiegender Sicherheit angenommen werden, dass die Daten tatsächlich nutzlos sind. Andernfalls müsste der Datentreuhänder selbst die Mechanismen zur Auswertung der Daten besitzen. Wie in 2.2 genannt wurde, besitzen eine Vielzahl an Datentreuhändern die Möglichkeit eine Qualitätssicherung der Daten zu durchzuführen. Dadurch kann ein Datentreuhänder die Qualität der Daten begründet einschätzen und entscheiden, ob der Datennutzende mit seiner Behauptung recht behält. Sollte der Datentreuhänder keine Qualitätssicherung liefern können oder sich bei der Beurteilung unsicher sein, wird im Zweifel für den Datengebenden entschieden.

4. **Fehlerhafte Coins.** Ein Streitfall aufgrund fehlerhafter Coins kann sowohl in Schritt 11 als auch Schritt 13 eintreten. Im Fall von Schritt 11, hat der Datengebende gerade seine Bezahlung aus dem Coinpostfach des Datentreuhänders abgeholt. Daraufhin hat er die partiell blinde Signatur des Exchanges überprüft und festgestellt, dass die für den Coin angegebene Signatur nicht valide ist. Um den Datentreuhänder darüber aufzuklären, leitet er den $sharedKey$, den $referenceCode$ des Postfacheintrags und den public Key des Aufrufs weiter und lässt den Datentreuhänder die Signatur prüfen. Wenn der Datentreuhänder anerkennt, dass die abgelegte Signatur ungültig ist, kann er den angegebenen public Key mit der Liste an Aufrufen abgleichen und den zu dem Aufruf gehörenden Datennutzenden bestrafen.

   Ein Coin kann genauso erst in Schritt 13 als fehlerhaft erkannt werden. Sollte die Signaturüberprüfung durch den Datengebenden in Schritt 11 ohne Probleme ablaufen, sendet er die Coins anschließend an den Exchange. Dieser wiederholt das Prüfen der Signatur. Da diese Validierung bereits in Schritt 11 reibungslose ablief, ist davon auszugehen, dass auch hier keine anderen Ergebnisse entstehen. Trotzdem kann ein Coin erst beim Einlösen an dem Exchange auf doppelte Ausgebung überprüft werden. Sollte der Exchange feststellen, dass ein übermittelter Coin zu einem vorherigen Zeitpunk schon einmal eingelöst wurde, so verweigert er die Annahme und meldet dies dem Datengebenden zurück. Daraufhin kann er genauso verfahren wie nach der Beschreibung für Schritt 11.

5. **Überschreitung des Bezahlzeitraums.** Wie schon in 4.7.3 erläutert hat der Datengebende nach einer Wartezeit von 12 Stunden die Möglichkeit eine Beschwerde beim Datentreuhänder einzureichen. Dafür speichert er sich beim Absenden eines Eintrags zum Datenpostfach die Zeit und fragt in beliebigen Abständen das Coinpostfach nach dem $referenceCode$ ab. Sollte 12 Stunden nach absenden des Eintrags noch keine Antwort vorliegen, so leitet er den $sharedKey$ zusammen mit dem Datenpostfacheintrag an den Datentreuhänder. Dieser validiert, dass der Datengebende tatsächlich seine Daten geteilt hat und prüft selbst ob bereits ein Eintrag unter dem $referenceCode$ des Eintrags eingesendet wurde. Trift dies nicht zu wird das Handeln des Datennutzenden als Verstoß gegen die Einhaltung des Bezahlzeitraums gewertet und entsprechend bestraft.

4.8 Reputationsvergabe

4.8.1 Ablauf

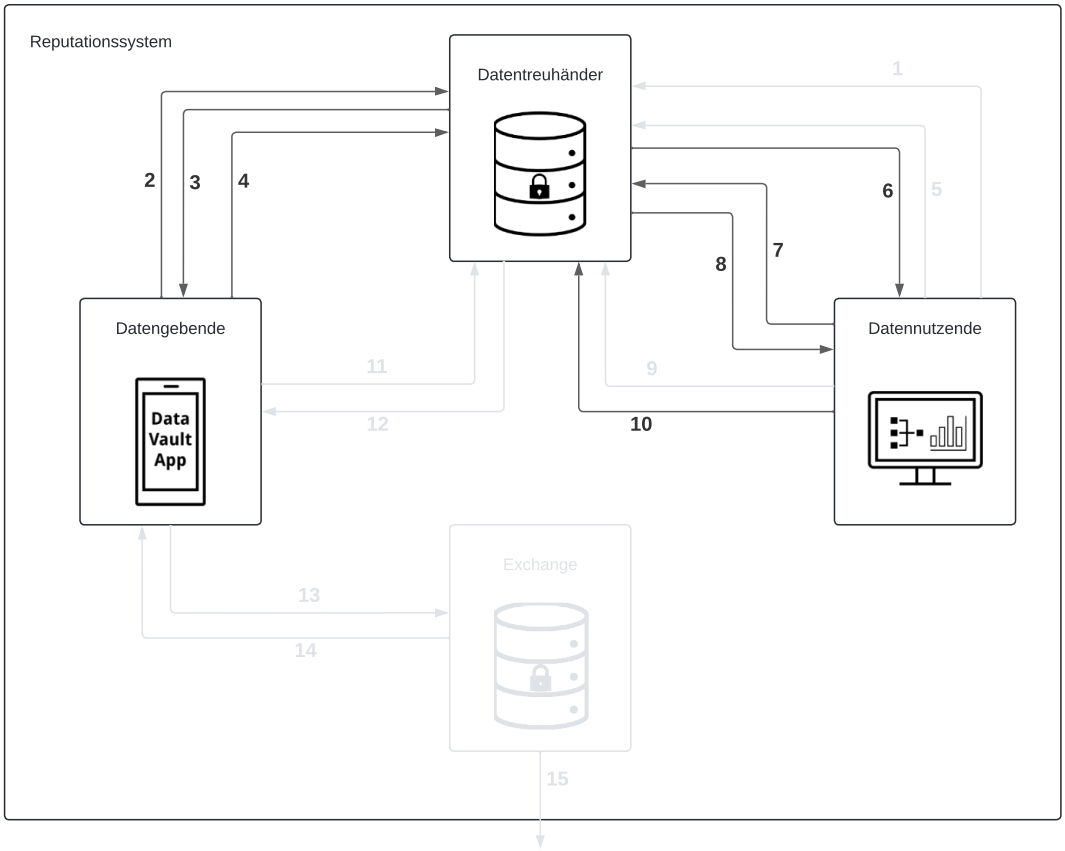

Abbildung 4.3: Reputationssystem Ablauf

5 Implementation

5.1 TRESOR Projekt

6 Auswertung

Nun da die Formalisierung des Konzepts abgeschlossen ist, kann mit der Auswertung von diesem begonnen werden. Dafür wird das Konzept hauptsächlich bezüglich der Sicheheit und des Rechenaufwands analysiert. Für die Analyse der Sicherheit werden eine Reihe von Angreifermodellen definiert und anhand deren gezeigt, welche Schwachstellen das Konzept gegen verschieden Angreifer aufweist. Zur Analyse des Rechenaufwands werden gemessene Laufzeiten der Implementation begründet und miteinander verglichen. Später werden diese Laufzeiten mit anderen Bezahlsystemen verglichen, um eine Einschätzung über die Effizienz verglichen mit heutigen Stardants zu liefern.

6.1 Bewertung der Sicherheit

Die Sicherheit eines Systems in Bezug auf verschiedene Angriffe ist ein essenzieller Faktor für den Erfolg dieses Systems. Insbesondere bei kryptographischen Protokollen wie Keyexchange Protokollen oder dem in 4 präsentierten Bezahlsystem ist die Sicherheit einer der entscheidenen Punkte die den Erfolg bestimmen können. Um einen Standard für solch eine Bewertung der Sicherheit einzuführen, definierten D. Dolev und A. Yao im Jahr 1983 das erste Angreifermodell [DY83]. Ein Angreifermodell beschreibt einen theoretischen Angreifer der probiert dem zu untersuchenden System Schaden zu zufügen oder Informationenen zu extrahieren. Dafür müssen die Rahmenbedingungen des Angreifermodells genau bestimmt werden. Es umfasst in der Regel Angreiferziele, Angreiferannahmen und Angreiferfähigkeiten [DMC19].

- **Angreiferziele** Für einen Angreifer gibt es mehrere Ziele die er mit seinem Angriffen erreichen können möchte. Er kann bspw. interesse daran haben, die übertragenen Informationen mitzuscheiden, Geheimnisse wie private oder Sitzungsschlüssel herauszufinden oder die Kommunikation vollständig zu verhindern.

- **Angreiferannahmen** Sie beschreiben das Umfeld und die Ressourcen des Angreifers. Häufig auftretende Annahmen sind die Unterscheidung zwischen externem oder internem Zugriff auf ein Netzwerk, die Beschränkung auf einen polynomial probabilistischen Angreifer, was beschreibt dass der Angreifer nur $O(n^k)$ Rechenaufwand bei einem Konstanten k und Eingabelänge n hat und bei der Berechnung zufällige Werte raten darf.

- **Angreiferfähigkeiten** Die Fähigkeiten eines Angreifers sind durch seine Handlungsmöglichkeiten definiert. So kann ein aktiver Angreifer die Fähigkeiten besitzen, Botnetzwerke zu verwenden um DDoS Attacken zu starten, per Brute Force probieren ein Passwort zu knacken oder Analysen des Geheimtexts durchführen, die Schlüsse auf die übertragenen Nachricht zulassen. Passive Angreifer hingegen haben meist nur die Möglichkeit die Kommunikation zu beobachten und Schlüsse aus dem Netzwerkverkehr anzuleiten.

Zusammen ergibt die Definition des Angreifermodell eine maximale Stärke die ein Angreifer besitzen kann und trotzdessen das System nicht brechen kann. Sollte er in nur wenigen Punkten mehr Macht besitzen als das Angreifermodell vorgibt, so ist die Sicherheit des Systems nicht mehr gewährleistet und der Angreifer kann sein definiertes Ziel erreichen. Daher spricht die Aufstellung eines starken Angreifermodells und die Verteidigung gegen dieses für die Robustheit des Systems. Eine genaue Definition des Angreifermodells ist wichtig, da unpräzise Formulierung ein mehrdeutiges Angreifermodell liefern, dass nicht genau bestimmen kann wogegen das System geschützt ist.

6.1.1 Angreifermodelle der Coin Generierungsphase

Bei einer Durchführung des Coin Generierungsprotokolls, treten Exchange und Datennutzender als Akteure auf. Für jeden von ihnen muss ein Angreifermodell definiert werden, um die Sicherheit bei korrumpierten Kommunikationteilnehmern zu zeigen. Zusätzlich soll die Sicheheit gegen einen außenstehenden Angreifer gezeigt werden.

Bösartiger Datennutzender Ein bösartiges Verhalten eines Datennutzenden kann durch eine Übernahme eines Datennutzenden durch einen externen Angreifer oder durch bösartige Interessen des Datennutzenden selbst entstehen. In beiden Fällen ist das Ziel des Angreifenden entweder Coins mit größerem monetären Wert zu erhalten als der ursprüngliche Zahlungseingang beim Exchange zulässt oder eine Möglichkeit zu erhalten selbst Coins zu signieren. Die dafür zu treffenden Annahmen sind, dass es sich um einen aktiven Angreifer handelt. Er verfügt über unbegrentzten Speicher und kann Algorithmen mit polynomieller probabilistischer Rechenzeit ausführen. Zusätzlich kennt er den public Key des Exchanges und den für den Zahlungseingang verwendeten apk. Mögliche Angriffe die er starten kann sind Replay-Attacken, welche eine zuvor gesendete Nachricht wiederholen, sowie Chosen-Plaintext-Attack, bei denen der Angreifer den Nachrichteninhalt bestimmt und observieren kann wie sich der Geheimtext verhält ‚Brute-Force-Attacken zum erraten des privaten Schlüssels und Geheimtextanalysen zur herausarbeitung des privaten Schlüssels.

Da der Ablauf der Zahlungseingang in Schritt 1 hauptsächlich von den konkret verwendeten Methode abhängt, kann hier davon ausgegangen werden, dass der Angreifer eine reguläre Transaktion leistet, da er ohne den damit entstehenden $CoinGenerationToken$ das Protokoll nicht vorführen kann. Anschließend erfragt er beim Exchange alle Token mit dem verwendeten apk. Nach Erhalt des soeben erstellten Token, kann er probieren Coins zu generieren, die den ES Wert des Token übersteigen und den $value$ dieser zusammen mit dem Token zu Exchange zu senden. Da der Exchange zuerst den summierten $value$ Betrag aller Coins mit dem ES des Tokens vergleicht, wird der Exchange bereits hier feststellen, dass ein Angreifer probiert mehr Geld zu erhalten als er bezahlt hat. Der Versuch schlägt hier fehl.

Der Angreifer ist daher gezwungen, eine zum ES passende Menge an $values$ zu generieren und zu übertragen. Der Exchange antwortet mit einer Liste von a, b für jeden $value$. Nun kann der Angreifer sich entscheiden, das Protokoll für partiell Blinde Signaturen (3.8) zu verlassen und anstatt die vorgesehen Berechnung für e auszuführen, e als Variable für den Chose-Plaintext-Angriff zu verwenden. Er schreibt jedem e einen leicht veränderten Wert zu, sendet diesen an den Exchange, erhält für jedes e ein Tupel aus (r, c, s, d) und kann probieren die unterschiede zwischen den verschiedenen r mit Hilfe von Geheimtextanalyse herauszuarbeiten. Da der Exchange zur Berechnung von r zwei dem Angreifer unbekannte Variablen benutzt (u, x) und auf das Ergebnis $mod\ q$ anwendet, ist bei einer ausreichend großen Wahl von u, x, q anzunehmen, dass der Ergebnis pseudozufällig erscheint und keine Hinweise auf x liefert. Dabei verliert der Angreifer zusätzlich bei jedem Versuch den Wert des Coins, da die entstehende Signatur nicht gültig ist.

Sollte der Angreifer e wie vom Protokoll vorgesehen berechnen und nach Erhalt von (r, c, s, d) probieren u, X zu erraten, sodass die Ergebnisse seiner Berechnung mit (r, c, s, d) übereinstimmen, so kann er diesen Brute-Force-Angriff ohne Kosten (abgesehen von Rechenaufwand) starten. Solange q groß genug gewählt (in der Implementierung 128bits) mit $u \in_R \mathbb{Z}_q$ ist die Change allein u korrekt zu bestimmen bereits $\frac{1}{2^{128}} \approx \frac{1}{3{,}4 \cdot 10^{38}}$ ausreichend gering, um das Gelingen des Erratens von u und x als vernachlässigbar anzunehmen.

Der Angreifer kann auch probieren anhand des public Keys des Exchange einen Faktorisierungsalgorithmus laufen zu lassen, um den private Key zu bestimmen. Noch sind aber keine Algorithmen bekannt, die das Faktorisierungsproblem in aufbringbarer Zeit effizient lösen können, weshalb der Angreifer auch mit diesem Vorgehen nicht an den private Key kommt. proof

Nach Ausstellung der Signatur ist der Wert jedes Coins fest zugeteilt. Wenn ein Angreifer versucht den $value$ des signierten Coins zu ändern, werden damit die Grundlagen zum Überprüfen der Signatur umdefiniert, was dazu führt, dass jede Validierung fehlschlägt. Somit verliert der Coin schlagartig seinen Wert, sobald der Angreifer den $value$ verändert.

Letztlich kann der Angreifer probieren einen Replay-Angriff zu starten und einen bereits einmal eingesetzten $CoinGenerationToken$ erneut mit einer Reihen an $values$ zum Exchange zu senden. Jedoch ist in 4.6 definiert, dass der Exchange jeden erhaltenen $CoinGenerationToken$ auf eine doppelte Einlösung überprüft. Daher wird der Exchange die Annahme des Tokens verweigern und der Angreifer schafft es auch hier nicht, mehr Coins zu erhalten als ihm zustehen.

Neugieriger Exchange Der Exchange stellt in der Coin Generierungsphase ein Rolle mit großer Verantwortung da. Er hat die Aufgabe Geldsummen sicher zu verwalten und trotz der Anonymität des Konzeptes genau die passenden Mengen ein- und auszuzahlen. Trotzdem wird das Angreifermodell für den Exchange hier nur schwach definiert. Dies hat vor allem den Hintergrund, dass das Konzept eines Exchanges zum Tausch von Geld in die jeweilige Kryptowährung bereits etabliert ist und viele Sicherheitsmaßnahmen bestehen, die den Exchange zu einem gutmütigen Verhalten zwingen [Bur+16][KL18][BDF21]. Deswegen wird im Folgenden nur die Sicherheit des Protokoll erwiesen aber nicht die Sicherheit des Exchanges. Angenommen der Exchange kann nicht durch unbefugte übernommen werden, so sind seine Angreiferziele, in Erfahrung zu bringen, welcher Datennutzende wie viele Coins besitzt und wie viel diese wert sind. Zusätzlich ist er daran interessiert herauszufinden, mit wem der Datennutzende der sich die Coins signieren lässt, handelt. Dafür möchte er sowohl die Identität des Datennutzende als auch des Coins wiedererkennen können und verschiedene signiervorgänge untereinander verlinken können. Die Angreiferannahmen beschränken sich zum größten Teil auf passives Verhalten. Er hält sich überwiegend an das Protokoll und beobachtet die Kommunikation mit dem Datennutzende, um daraus Informationen zu sammeln. Seine Rechenleistung ist polynomial probabilistisch und er kennt sämtliche Informationen die im ersten Schritt angegeben werden. Die Angreiferfähigkeiten sind ebenso eingeschränkt. In den meisten Fällen antwortet er auf Anfragen des Datennutzende wie das Protokoll vorgibt. Nur die Wahl der Berechnungswerte für die partiell blinden Signatur kann er variieren.

Wie definiert, hält sich der Exchange zunächst an das Protokoll. Er erstellt beim Eintrefen einer Zahlung einen $CoinGenerationToken$ mit dem ES und apk gleich den Werten der Zahlung und legt diesen mit dem apk verschlüsselt in seinem Speicher ab. Da der apk ein regulärer public Key ist, kann der Exchange keine fundierten Kenntnissen aus einem apk ziehen. Erst sobald der selbe apk für eine spätere Zahlung erneut angegeben wird, kann davon ausgegangen werden, dass der Zahlungseingang von der gleichen Person stammt. Wenn ein Datennutzender aber für jedes Zahlung einen neuen apk generiert, kann der Exchange zwei verschiede Zahlung nicht zu der gleichen Person zurückführen.

Auf Anfrage gibt er den $CoinGenerationToken$ heraus. Bei dem berechnen der partiell blinden Signatur ist der einzige Wert, für den eine Abweichung von Protokoll sinnvoll sein kann, der private Schlüssel. Die Werte u, s, d, z, c, r sind bei jedem signierten Coin unterschiedlich. Diese gleich zu behalten würde ab dem zweiten Coin dem Datennutzenden auffallen. Der private Schlüssel hingegen ist dem Datennutzenden nicht bekannt und die Auswirkung beim auswechseln des privaten Schlüssel sind nicht feststellbar. Daher kann der Exchange probieren, für jeden signierten Coin einen eigenen privaten Schlüssel zu verwenden. Dadurch kann er beim späteren Erhalt von Coins im Bezahlsystem durch den Datengebenden nachvollziehen, aus welcher Signieranfrage der Coin stammt und kann so die Beziehung zwischen dem Datennutzenden und Datengebenden vermuten. An dieser Stelle greift jedoch der $BDLEQ$, welcher in 3.5 erklärt wurde und verhindert, dass der Exchange ungehindert den Schlüssel austauschen kann. Bei einem Versuch den Schlüssel zu ersetzen, zeigt der $BDLEQ$, dass

der verwendete Schlüssel nicht mehr mit dem beworbenen übereinstimmt. Somit ist der Exchange gezwungen den gleiche privaten Schlüssel einzusetzen, was ihm die potenzielle Wiedererkennung im späteren Verlauf verhindert.

Angreiferziele: - will wissen welcher DN wie viele coins hat und die viel geld - will verschiedene signiervorgänge verlinken können Angreiferannahmen: - passive beobachtend - poly rechenaufwand - kennt werte des coingentoken - hält sich ans protokoll außer immer gleichen sk Angreiferfähigkeiten: -

erhält zahlung -> stellt token aus (apk kann selbst bei gleichen DN jedes mal anders sein) -> kein Informationsgewinn aus apk antwortet mit token -> erhält values -> benutzt für jede anfrage unterschiedliches keypair -> bdleq stimmt nicht mehr -> DN beschwert sich und meldet exchange -> wird gebannt muss also gleich schlüssel benutzen -> kennt nur value aber nicht nonce -> wenn token eingelost wird ist nonce unbekannt und value stammt aus kleiner menge an werten -> also auch viel zu häufig vorkommend um schlüsse zu ziehen Kann die Identität auf keiner weise nachverfolgen

Außenstehender Angreiferziele: - will selbst coins erhalten - will DN vom erhalten seiner coins abhalten Angreiferannahmen: - kennt DN-ID des DN - kenn den apk und kann jeden apk zu dn-id matchen - unbegrentz speichplatz und polynialzeit Rechenleistung - ist aktiv Angreiferfähigkeiten: - chosen-plaintext - brute force - cryptoanalysis

Kann bei schritt 2 sich als dn ausgeben und nach token für apk fragen. Erhält verschlüsselten token -> polynomialzeit reicht nicht, cryptoanalysis liefert keine herausstechenden ergebnisse kann probieren token zu erraten -> $100.000 * 2^2 56$ -> $1/1,15 * 10^8 2$-> vernachlässigbar kann nichts weiter tuhen da partblindsig über gleiche tcp sein muss und token zum start nötig ist wiederholtes ausprobieren von token kann maximal dos verursachen

6.1.2 Angreifermodelle der Bezahlphase

Für die Definition der Angreifermodelle wird zunächst angenommen, dass jeder Akteur der an dem System beteiligt ist, mehr oder weniger aktiv an der Ausnutzung möglicher Schwachstellen interessiert ist. Somit ergeben sich für die 4 verschiedenen Akteure 4 Angreifermodelle mit unterschiedlichen Eigenschaften.

- **Bösartige Datennutzende**
- **Bösartige Datengebende**
- **Ehrlich aber neugieriger Datentreuhänder**
- **Ehrlich aber neugieriger Exchange**

6.2 Setup / Implementation

6.3 Analyse des Rechenaufwands

6.3.1 Metrics

7 Bewertung

7.1 Erfüllen der Anforderungen

7.2 Wie kann ein privatsphäreschützender Anreiz zur Benutzung eines Datentreuhändermodells geschaffen werden?

7.3 Wie kann dieser Anreiz gegen Missbrauch geschützt werden?

7.4 (Discussion)

8 Begrenzungen

9 Zusammenfassung

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang Software-System-Entwicklung selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel — insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen — benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe.

Hamburg, den 1. Oktober 2024

Knut Hoffmeister

Literatur

[] *What is a cryptocurrency transaction blockchain confirmation?* URL: https://www.bitcoin.com/get-started/what-is-a-confirmation (besucht am 12. 09. 2024).

[AAJ24] Md Sadi Arman, Shamim Al Mamun und Nuray Jannat. *A modified AES based approach for data integrity and data origin authentication*. In: *2024 3rd International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE)*. IEEE. 2024, S. 1–6.

[aka] akamai. *Was ist ein CDN (Content Delivery Network)?* URL: https://www.akamai.com/de/glossary/what-is-a-cdn#:~:text=Bei%20einem%20CDN%20handelt%20es,die%20auf%20die%20Inhalte%20zugreifen. (besucht am 21. 08. 2024).

[AO00] Masayuki Abe und Tatsuaki Okamoto. *Provably secure partially blind signatures*. In: *Advances in Cryptology—CRYPTO 2000: 20th Annual International Cryptology Conference Santa Barbara, California, USA, August 20–24, 2000 Proceedings 20*. Springer. 2000, S. 271–286.

[Ara16] Bálint Aradi. *Einführung in die Gruppentheorie*. 2016.

[Bao22] Jiaxu Bao. *Research on the security of elliptic curve cryptography*. In: *2022 7th International Conference on Social Sciences and Economic Development (ICSSED 2022)*. Atlantis Press. 2022, S. 984–988.

[BDF21] Carsten Baum, Bernardo David und Tore Kasper Frederiksen. *P2DEX: privacy-preserving decentralized cryptocurrency exchange*. In: *International Conference on Applied Cryptography and Network Security*. Springer. 2021, S. 163–194.

[Ber06] Daniel J Bernstein. *Curve25519: new Diffie-Hellman speed records*. In: *Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9*. Springer. 2006, S. 207–228.

[Bla+20] Aline Blankertz u. a. *Datentreuhandmodelle-Themenpapier*. In: (2020).

[Bog08] Oleg Vladimirovič Bogopolskij. *Introduction to group theory*. Bd. 6. European Mathematical Society, 2008.

[BS21a] Aline Blankertz und Louisa Specht. *Wie eine Regulierung für Datentreuhänder aussehen sollte*. In: *Policy-Brief. Juli. Berlin: Stiftung Neue Verantwortung eV https://www.stiftung-nv.de/sites/default/files/regulierung_fuer_datentreuhaender.pdf* (2021).

[BS21b] ALINE BLANKERTZ und DR LOUISA SPECHT-RIEMENSCHNEIDER. *Neue Modelle ermöglichen*. In: (2021).

[Bun] Bunderdruckerei. *Datenblatt-Datentreuhaender*. URL: https://www.bundesdruckerei-gmbh.de/files/dokumente/pdf/datenblatt-datentreuhaender.pdf (besucht am 03. 05. 2024).

[Bun21] Die Bundesregierung. *Datenstrategie der Bundesregierung*. In: *Die Bundesregierung, Berlin* (2021).

[Bur+16] Jeffrey Burdges u. a. *Enabling secure web payments with GNU Taler*. In: *Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings 6*. Springer. 2016, S. 251–270.

[But+13] Vitalik Buterin u. a. *Ethereum white paper*. In: *GitHub repository* 1 (2013), S. 22–23.

[But+20] Vitalik Buterin u. a. *Combining GHOST and casper*. In: *arXiv preprint arXiv:2003.03052* (2020).

[Cha83] D Chaum. *Blind signatures for untraceable payments*. 1983.

[Coi] CoinMarketCap. *Ranking der größten Kryptowährungen nach Marktkapitalisierung im September 2024*. URL: https://de.statista.com/statistik/daten/studie/296205/umfrage/marktkapitalisierung-digitaler-zahlungsmittel/ (besucht am 24. 09. 2024).

[Dav+18] Alex Davidson u. a. *Privacy pass: Bypassing internet challenges anonymously*. In: *Proceedings on Privacy Enhancing Technologies* (2018).

[Dig23] Digiconomist. *Bitcoin average energy consumption per transaction compared to that of VISA as of May 1, 2023*. Graph. 2023. URL: https://www.statista.com/statistics/881541/bitcoin-energy-consumption-transaction-comparison-visa/ (besucht am 12. 09. 2024).

[DMC19] Quang Do, Ben Martini und Kim-Kwang Raymond Choo. *The role of the adversary model in applied security research*. In: *Computers & Security* 81 (2019), S. 156–181.

[DY83] Danny Dolev und Andrew Yao. *On the security of public key protocols*. In: *IEEE Transactions on information theory* 29.2 (1983), S. 198–208.

[Fac24] Facebook. *Umsatz von Facebook weltweit nach Segmenten in den Jahren von 2009 bis 2023*. 2024. URL: https://de.statista.com/statistik/daten/studie/151159/umfrage/umsatz-von-facebook-in-2009-nach-segmenten/ (besucht am 27. 03. 2024).

[FK20] Oliver Falck und Johannes Koenen. *Rohstoff"Daten: Volkswirtschaftlicher Nutzen von Datenbereitstellung-eine Bestandsaufnahme*. 113. ifo Forschungsberichte, 2020.

[Goo24] Alphabet; Google. *Alphabet, Werbeumsätze von Google in den Jahren 2001 bis 2023*. 2024. URL: https://de.statista.com/statistik/daten/studie/75188/umfrage/werbeumsatz-von-google-seit-2001/ (besucht am 27. 03. 2024).

[Gre13] Matthew Green. *The Many Flaws of Dual_EC_DRBG*. 2013. URL: https://blog.cryptographyengineering.com/2013/09/18/the-many-flaws-of-dualecdrbg/ (besucht am 10. 07. 2024).

[GS11] Kamlesh Gupta und Sanjay Silakari. *Ecc over rsa for asymmetric encryption: A review*. In: *International Journal of Computer Science Issues (IJCSI)* 8.3 (2011), S. 370.

[Ham22] Universität Hamburg. *Tresor Projekt - About*. 2022. URL: https://tresor-projekt.de/about/ (besucht am 28. 02. 2024).

[han17] handysektor.de. *Geheimnis Staumelder: Wie funktioniert eigentlich Google Maps?* 2017. URL: https://www.handysektor.de/artikel/geheimnis-staumelder-wie-funktioniert-eigentlich-google-maps (besucht am 07. 08. 2024).

[Har18] Jack Hardinges. *What is a data trust?* 2018. URL: https://theodi.org/insights/explainers/what-is-a-data-trust/?saved#cookies-form (besucht am 15. 04. 2024).

[JK16] Tun-Min Catherine Jai und Nancy J King. *Privacy versus reward: Do loyalty programs increase consumers' willingness to share personal information with third-party advertisers and data brokers?* In: *Journal of Retailing and Consumer Services* 28 (2016), S. 296–303.

[KB21] Jürgen Kühling und Benedikt Buchner. *Datentreuhänder*. 2021.

[Kee21] Lila Kee. *RSA is dead - We Just Haven't Accepted It Yet*. 2021. URL: https://www.forbes.com/sites/forbestechcouncil/2021/05/06/rsa-is-dead---we-just-haventaccepted-ityet/ (besucht am 24. 06. 2024).

[KL18] Chang Yeon Kim und Kyungho Lee. *Risk management to cryptocurrency exchange and investors guidelines to prevent potential threats*. In: *2018 international conference on platform technology and service (PlatCon)*. IEEE. 2018, S. 1–6.

[LM10] Manfred Lochter und Johannes Merkle. *Elliptic curve cryptography (ECC) brainpool standard curves and curve generation*. Techn. Ber. Independent Submission, 2010.

[Mil85] Victor S Miller. *Use of elliptic curves in cryptography*. In: *Conference on the theory and application of cryptographic techniques*. Springer. 1985, S. 417–426.

[ML13] Johannes Merkle und Manfred Lochter. *Elliptic curve cryptography (ECC) brainpool curves for transport layer security (TLS)*. Techn. Ber. Internet Engineering Task Force (IETF), 2013.

[MY18] Dindayal Mahto und Dilip Kumar Yadav. *Performance analysis of RSA and elliptic curve cryptography.* In: *Int. J. Netw. Secur.* 20.4 (2018), S. 625–635.

[Nak08] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. In: *Decentralized business review* (2008).

[PLS14] Ronald Petrlic, Sascha Lutters und Christoph Sorge. *Privacy-preserving reputation management*. In: *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. 2014, S. 1712–1718.

[Pro13] Malte Stöfen Prof. Dr. Volker Beeck Prof. Dr. Thilo Seyfriedt. *Treuhandschaft*. 2013. URL: https://wirtschaftslexikon.gabler.de/definition/treuhandschaft-47435/version-270699 (besucht am 22. 04. 2024).

[Ric] Frederik Richter. *DDV-Webtalk Datentreuhänder - Modelle*. URL: https://youtu.be/10UbvTOrjiM?si=bg-4xdCNRH9b93-D (besucht am 03. 05. 2024).

[Sch24] Peter Schöllauf. *What is Bitcoin? Complete Beginner's Guide [2024]*. 2024. URL: https://www.blockpit.io/blog/what-is-bitcoin (besucht am 12. 09. 2024).

[Sul13] Nick Sullivan. *A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography*. 2013. URL: https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography (besucht am 17. 06. 2024).

[TEA24] THE INVESTOPEDIA TEAM. *What Is Ethereum and How Does It Work*. 2024. URL: https://www.investopedia.com/terms/e/ethereum.asp (besucht am 12. 09. 2024).

[TRE24] TRESOR. *Ergebnisse der Nutzerstudie zur Anforderungsdefinition*. 2024. URL: https://tresor-projekt.de/news/user-study-results/ (besucht am 30. 08. 2024).

[YCH] YCHARTS. *Ethereum Average Block Time (I:EBT)*. URL: https://ycharts.com/indicators/ethereum_average_block_time (besucht am 12. 09. 2024).

[ZDF23] ARD; ZDF. *Anzahl der Internetnutzer in Deutschland in den Jahren 1997 bis 2023*. 2023. URL: https://de.statista.com/statistik/daten/studie/36146/umfrage/anzahl-der-internetnutzer-in-deutschland-seit-1997/ (besucht am 07. 08. 2024).