# DashPay als Beispiel für ein dezentrales, nutzerfreundliches Zahlungssystem

Marc Herschel

Hochschule Hannover, Ricklinger Stadtweg 120, 30459 Hannover, Germany
marc@herschel.io
https://hs-hannover.de

**Abstract.** Seit den letzten Jahren steigt die Adaption von Kryptowährungen durch die allgemeine Bevölkerung stetig. Eine hohe technische Einstiegsschwelle bleibt jedoch für den durchschnittlichen Nutzer bestehen, um erfolgreich und sicher Transaktionen durchzuführen. Um die Massenadaption von Kryptowährungen weiterhin voranzutreiben, ist es notwendig, eine einfachere und fehlerfreiere Variante zum durchführen von Transaktionen als momentan über gewöhnliche Wallets möglich ist anzubieten. Das Blockchain-Projekt Dash entwickelt momentan mit Dash Platform einen Ansatz zum Realisieren von dezentralisierten Anwendungen. Im Gegensatz zu bekannteren Projekten wie z. B. Ethereum, in dem Smart Contracts durch Programmcode auf der Blockchain realisiert sind, setzt Dash auf ein datenorientiertes Vorgehen (Data Contracts). Dadurch erfolgt eine Auslagerung der Anwendungslogik auf die Klienten, die Anwendungsdaten hingegen werden in einer separaten Blockchain gespeichert. In dieser Ausarbeitung wird anhand von DashPay die Technologie hinter der Dash Platform vorgestellt. Ebenso werden die elementaren Grundbausteine, die DashPay ermöglichen wie z. B. Identitäten (Identities) oder der Dash Platform Name Service (DPNS) behandelt. DashPay ist ein Data Contract in dem sich Nutzer eindeutig durch Identitäten ausweisen und sich somit Kontaktanfragen zuschicken können. Kontakte können dadurch bidirektionale, direkte Zahlungskanäle untereinander aufbauen. DashPay ermöglicht somit einfache, schnelle und sichere Zahlungen über das Dash-Netzwerk. Nutzer von DashPay müssen für Zahlungen durch die bidirektionalen Kontakte keine langen und unübersichtlichen öffentlichen Adressen mehr untereinander austauschen.

**Keywords:** Blockchain, Dash, DashPay, Dash Platform, Dezentralisierte Anwendungen, Kryptowährungen

## 1 Motivation

Seit den letzten Jahren steigt die Adaption von Kryptowährungen durch die allgemeine Bevölkerung stetig. Die Marktkapitalisierung des gesamten Kryptomarkts ist demzufolge laut CoinMarketCap[14] im Zeitfenster zwischen dem 13.04.2017 bis zum 13.04.2021 um etwa 7000% angestiegen. Mit einer derzeitigen Marktkapitalisierung (Stand: 13.04.2021) von knapp 1,8 Mrd. Euro reiht

sich die gesamte Assetklasse der Kryptowährungen also nur knapp hinter der Marktkapitalisierung der Apple-Aktie[16](Stand: 13.04.2021) ein. Falls also eine Massenadaption in den nächsten Jahrzehnten stattfinden sollte, existiert hier eine große Wachstumschance. Denn laut einer Kaspersky-Umfrage[30] von 2019 verstehen gerade einmal 10% der Probanden, wie Kryptowährungen tiefergehend funktionieren.

Auch im Jahre 2021 sind Kryptowährungen für den durchschnittlichen Verbraucher schwer verständlich und unbequem zu benutzen. Die durch einen öffentlichen Schlüssel des Wallets abgeleiteten langen und kryptischen Adressen sorgen hier nicht für mehr Verständnis. Der Austausch solcher Adressen bietet schnell einen Angriffsvektor durch z.B. einen kompromittiertes Endnutzergerät oder Fehler beim Austausch zwischen den Zahlungsteilnehmern. So ist bei größeren Transaktionen noch immer erst einmal ein Testlauf mit einer kleinen Summe empfehlenswert, um einen Totalverlust durch eine falsche Adresse zu vermeiden. Wenn man hier nun einen Vergleich zum klassischen Bankensystem oder auch beliebten Zahlungsanbietern wie PayPal[22], CashApp[1] oder Stripe[28] zieht, fällt auf, dass Kryptowährungen hier noch einiges an Nutzerfreundlichkeit aufzuholen haben, um eine ernst zu nehmende Alternative gegenüber den derzeitig überwiegend genutzten traditionellen zentralisierten Zahlungssystemen zu bieten. Das in dieser Ausarbeitung behandelte dezentralisierte Zahlungssystem DashPay[6] ermöglicht es, einen Konkurrenten zu schaffen und zumindest bezüglich der Nutzerfreundlichkeit mit Zahlungssystemen wie PayPal und etc. mitzuhalten.

DashPay[45] baut auf der Technologie von Dash Platform[33] auf. Dash Platform ist ein Projekt der Dash Core Group (DCG)[4] und baut wiederum auf dem bestehenden Netzwerk der Kryptowährung Dash[3] auf. Es handelt sich bei Dash Platform um einen Technologie-Stack für die Entwicklung von dezentralisierten Anwendungen. Im Gegensatz zu bereits etablierten Ansätzen bezüglich dezentralisierter Anwendungen wie z. B. Ethereum Smart Contracts[17], bei dem die Anwendungslogik als kompiliertes Programm auf der Blockchain residiert, werden bei Dash Platform lediglich die Anwendungsdaten eines Programmes über sogenannte Data Contracts in einer Blockchain gespeichert. Dafür gibt es eine Dash Platform Komponente mit dem Namen *Drive*, die als eine Art Dokumentendatenbank auf der Blockchain fungiert. Klienten können über die sogenannte Decentralized API (DAPI)[10], einer dezentralisierten API mit Dash Platform über JSON-RPC Endpunkte, die von beliebigen Dash Masternodes[20] zur Verfügung gestellt werden, kommunizieren.

Bei DashPay handelt es sich also um ein Data Contract, das als Protokoll für ein dezentralisiertes Zahlungssystem dient und es somit Entitäten mit unterschiedlichen Identitäten erlaubt, bidirektionale direkte Zahlungskanäle untereinander aufzubauen. Zusätzlich gibt es einige Features wie die Möglichkeit, Kontakte zu pflegen, es müssen also nicht mehr Kryptoadressen ausgetauscht werden, sondern es ist durch DashPay möglich, Transaktionen mit einem Kontakt

durchzuführen, wie es z. B. durch Benutzernamen bei PayPal.me[23] möglich ist. Im Rahmen dieser Ausarbeitung werden zuerst die notwendigen Grundlagen bezüglich Dash Platform und Identitäten erläutert. Anschließend wird die grundlegende Funktionsweise des DashPay Data Contract offengelegt und die technischen Aspekte hinter dem Design dieser dezentralisierten Anwendung konkretisiert. Zum Zeitpunkt der Ausarbeitung existiert bereits ein Testprogramm[8] für DashPay, das auf dem Dash-Testnetzwerk[29] operiert. Die derzeitigen Ergebnisse des Testprogramms werden ebenfalls miteinbezogen, um beispielhaft den momentanen Stand von DashPay näherzubringen und den praktischen Nutzen der hier beschriebenen theoretischen Grundlagen zu verdeutlichen und zu demonstrieren, warum DashPay als eine positiv zu betrachtende Kontribution bezüglich der Massenadaption von Kryptowährungen zu betrachten ist.

## 2 Grundlagen

Im Abschnitt der Grundlagen wird eine detailliertere Sicht auf die grundlegende Funktionsweise von Dash Platform gegeben. Ebenfalls werden Identitäten[40] kurz im Groben behandelt, da diese im Kontext von DashPay in Verbindung mit dem Dash Platform Name Service (DPNS)[39] benötigt werden, um die Vergabe von eindeutigen Nutzernamen für Entitäten zu ermöglichen.

### 2.1 Dash Platform

Dash Platform[32] ist ein Technologie-Stack, der es ermöglicht, dezentralisierte Anwendungen basierend auf dem Dash[31] P2P-Zahlungsnetzwerk zu entwickeln. Die beiden Hauptkomponenten sind hierbei Drive für das dezentralisierte Persistieren von Dokumenten und die Decentralized API (DAPI), die es Entwicklern ermöglicht, über JSON-RPC oder gRPC mit Dash Platform zu interagieren.

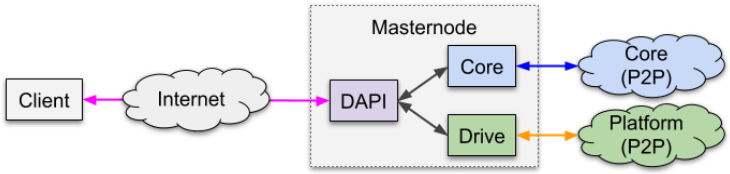


**Fig. 1.** Zugriff auf die Drive-Komponente über die DAPI mithilfe eines Masternodes[10]

In Abbildung 1 ist beispielhaft dargestellt wie ein Klient über die von einem Masternode[20] angebotene DAPI sowohl auf die Core-Blockchain, die von Dash für Transaktionen verwendet wird, als auch auf die separate Platform Chain[24], welche die Daten der Drive-Komponente persistiert zugreifen kann. Masternodes sind nutzergestellte Server im Dash Netzwerk, die Zusatzfunktionen wie z. B. die DAPI anbieten. Dadurch das jedes Masternode eine mit anderen Masternodes synchronisierte Kopie der Core und Platform Chain bereithält, ist so ein

ausfallsicherer und dezentralisierter Cloud-Speicherplatz gegeben. DAPI-Nutzer können also ein beliebiges Masternode ansteuern und erhalten immer die aktuellsten Informationen aus dem Netzwerk bezüglich der gestellten Anfrage. Masternodes werden gegen ein Pfand des Betreibers gestellt und bieten einen monetären Anreiz, den bestmöglichen Service für Netzwerkteilnehmer (Proof of Service) anzubieten.

Aus Gründen der Nutzerfreundlichkeit und Effizienz hat man sich für zwei separate Blockchains entschieden.[38] Die Platform Chain für die Drive Komponente läuft exklusiv auf dem Masternode-Netzwerk und benötigt somit keinen Proof of Work durch Miner. Stadtessen wird Byzantine Fault Tolerance[25] zur Validierung der Platform Chain verwendet. Durch diesen Ansatz ist es ebenfalls möglich, deterministische Transaktionsgebühren für Zustandsänderungen auf der Platform Chain zu garantieren, mit der Core Chain wäre dies nicht möglich, da die Gebühren dort abhängig von der Auslastung der Miner sind. Dezentralisierte Anwendungen sollen außerdem nutzerfreundlich und responsiv gestaltet werden können, durch das Masternode-Netzwerk sind somit schnellere Zustandsänderungen ($\leq$ 10 Sekunden) an der Platform Chain möglich als durch die Core Chain, in der eine Blockzeit von 2,5 Minuten existiert. Man stelle sich als Pro Platform Chain Argument einfach einen dezentralisierten Blog vor, in dem Kommentare im besten Falle erst nach 2,5 Minuten finalisiert sind.

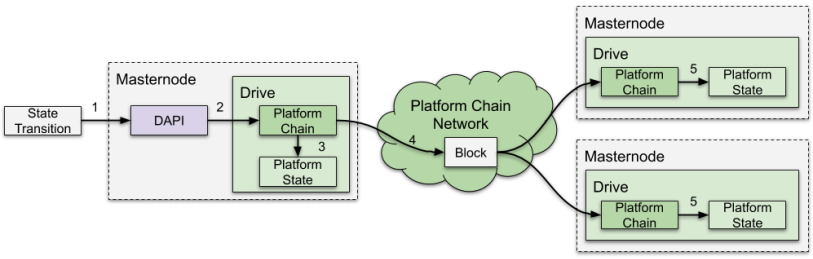


**Fig. 2.** Propagation einer State Transition über ein Masternode[13]

In Abbildung 2 ist zu erkennen, dass der globale Drive-Zustand über die verschiedenen Masternodes im Netzwerk verteilt liegt.[13] Um mit Drive zu interagieren, kommt das Dash Platform Protocol (DPP)[26] zum Einsatz. Dieses gewährt Integrität und Konsistenz der persistierten Daten. Das DPP besteht aus drei Komponenten: Data Contracts, Documents und State Transitions. Ein Data Contract ist ein Datenbankschema, welches von einer Entität (z. B. Entwicklern) über eine Identität innerhalb von Dash Platform registriert wird. Data Contracts werden anhand von JSON Schema[18] definiert. Um Speicherplatz zu sparen, findet außerdem eine Serialisierung in CBOR[2] statt. Nach der abgeschlossenen Registrierung wird das Data Contract bei jeder zugeordneten State Transition zur Validierung dieser verwendet.[9] Nur valide State Transitions werden in den globalen Zustand von Drive übernommen. Die eigentlichen Daten werden

anschließend als Dokument[11] im JSON-Format gespeichert. Dokumente sind immer einem Data Contract, das die Konsistenz des Dokumentes garantiert, zugeordnet und als Einträge in der Dokumentendatenbank zu sehen. Um die Datenbankoperationen `CREATE`, `UPDATE` und `DELETE` abzubilden, kommen State Transitions[27] zum Einsatz. State Transitions bestehen aus Änderungen an Dokumenten oder einem zu registrierenden Data Contract, einem Identifier zur Identifikation der dazugehörigen Anwendung und einer Signatur des Absenders für die Gewährleistung der Authentizität. In Abbildung 2 ist die Funktionsweise von State Transitions visualisiert. State Transitions werden von der Anwendungslogik über die DAPI an ein Masternode weitergeleitet und anschließend auf der Platform Chain geordnet, validiert und in einem Block gespeichert. Nur valide State Transitions fließen anschließend in den Platform State ein. Neue Blöcke werden an andere Masternodes im Netzwerk propagiert - diese aktualisieren dann die lokale Kopie der Platform Chain und des Platform States durch die neuen, validen State Transitions. Um den aktuellsten Platform State zu erreichen, reicht es aus, alle Blöcke der Platform Chain durch die Platform State Machine zu schicken. Der Platform State besteht aus allen registrierten Data Contracts und dem Application State, der sämtliche Dokumente in der aktuellsten Version beinhaltet. Sämtliche Inhalte des Platform State sind als validiert und integer zu betrachten.

### 2.2 Identitäten (Identities)

Identitäten (Identities)[40] legen den Grundstein[15] für Benutzernamen und somit für die einfache Nutzerinteraktion im Dash-Zahlungsnetzwerk. Identities bilden allerdings keine Benutzernamen direkt ab, dafür ist der Dash Platform Name Service (DPNS) notwendig. Eine Identity ist ein öffentlicher Schlüssel, der auf der Platform Chain gespeichert ist. Über eine Identity lässt sich also ein Benutzerprofil einer dezentralisierten Anwendung verwalten, da wie in Abschnitt 2.1 beschrieben zum Modifizieren von Dokumenten eine gültige Signatur des dazugehörigen Absenders notwendig ist. Um die Transaktionskosten einer State Transition zu zahlen, ist außerdem für jede Identity ein Betrag an gesperrten Dash (per *Lock Transaction*) auf der Core Chain hinterlegt. Um eine neue Identität zu erstellen, muss ein Nutzer ersteinmal die Kosten für das Hinterlegen des öffentlichen Schlüssels auf der Platform Chain tragen. Falls die hinterlegten Dash nach einer gewissen Anzahl von State Transitions aufgebraucht sind, ist es möglich den für Dash Platform dedizierten Kontostand neuaufzuladen. Identities bieten also einen Mechanismus, um Zahlungen für das Verwenden von Dash Platform (Layer 2, Platform Chain) über die Core Chain auf Layer 1 durchzuführen. Durch die Registrierung eines öffentlichen Schlüssels auf der Platform Chain ist es durch Identities ebenfalls möglich, Verschlüsselung und Message Authentication in dezentralisierten Anwendungen zu realisieren. Da für jede Registrierung einer Identity eine Transaktion zum Speichern des öffentlichen Schlüssels auf der Platform Chain notwendig ist, lässt sich jede Identität außerdem auf angewandten Proof of Work auf der Core Chain zurückführen.

### 2.3 Dash Platform Name Service (DPNS)

Um nun Identities mit DashPay-Benutzernamen zu verknüpfen, wird ein System zur Namensauflösung benötigt. Durch den Dash Platform Name Service (DPNS)[39] ist ein solches System durch ein Data Contract als dezentrale Anwendung basierend auf Dash Platform gegeben[21]. Nutzer oder Entwickler können durch den DPNS einen Namen für eine Identity registrieren und diese somit menschenlesbar darstellen. Dadurch das es sich bei einer Identity um einen öffentlichen Schlüssel abgespeichert auf der Platform Chain handelt, kann durch den zugehörigen privaten Schlüssel der Besitz eines registrierten Namens nachgewiesen werden und darauffolgend auch Vertrauen zwischen registrierten Namen aufgebaut werden.

Registrierte Namen sind im gesamten Kontext von Dash Platform als eindeutig unterscheidbar zu betrachten[39, 3.] Eine Kompatibilität zu DNS[12] wird gewährleistet, dadurch ist es möglich, Subdomains für bereits vorhandenen Namen zu registrieren. Im Kontext von DPNS ist das z. B. für personalisierte Benutzerprofile dezentralisierter Anwendungen von Relevanz (Im Falle von DashPay z. B. um im späteren Stadium des Projektes PayPal.me ähnliche URLs anzubieten)[39, 5.] Es ist für Nutzer nicht möglich, TLDs frei zu definieren, lediglich die Entität, welche das DPNS Data Contract registriert hat, ist in der Lage, neue TLDs einzuführen. Momentan sind nur Dash Identity Records unterstützt, um die Auflösung zwischen Name $\Leftrightarrow$ Identity zu gewährleisten.[39, 6.i] Zukünftig ist allerdings auch die Unterstützung von anderen Record-Typen wie URLs, arbiträren Daten oder Dash Data Contract IDs geplant. Die Registration eines Namens über DPNS findet durch zwei aufeinanderbauende Schritte (Preorder und Registration) statt[39, 6.v] Der Preorder-Schritt existiert, um durch die gegebene dezentrale Natur des DPNS mögliche Man-in-the-Middle-Angriffe zu verhindern. Hierbei wird der Name für die Identität registriert, ohne den eigentlichen Namen zu veröffentlichen. Der Nutzer generiert für das Preorder-Dokument einen Hashwert bestehend aus einem zufällig generiertem Salt und dem zu registrierenden Namen. Es ist für Drittparteien somit unmöglich, die Nachricht mit dem Namen abzufangen und vor dem Nutzer zu registrieren. Sobald das Preorder-Dokument dann im Platform State finalisiert ist, kann die Registrierung durch den Registration-Schritt abgeschlossen werden. In diesem Schritt wird der registrierte Name aufgedeckt und mit der Identität der registrierenden Entität verknüpft. Um zu beweisen, dass der Name wirklich von der Identität angefordert wurde, erfolgt eine Offenlegung des verwendeten Salt des ersten Schritts. Um die Gültigkeit der Registrierung zu validieren, reicht es aus, den Hash aus dem angeforderten Namen und dem übermittelten Salt zu bilden und mit dem Wert aus dem zuvor eingereichten Preorder-Dokument zu vergleichen.

DPNS ermöglicht es nach Namen und Identität aufzulösen, außerdem existiert eine Suchfunktion, um alle Namen mit einem gemeinsamen Präfix aufzulisten[39, 6.vi]. Im aktuellen Zustand ist die Freigabe und Modifikationen von bestehenden Einträgen nicht vorgesehen. Subdomains können aktuell nur von der besitzenden

Entität eines Namens registriert werden[39, 7]. Um mit DPNS aus einem Programm heraus zu interagieren, kann beispielsweise das Dash SDK[5] verwendet werden.

## 3 DashPay

Nachdem die Grundlagen der auf Dash Platform aufbauenden dezentralisierten Anwendungen abgearbeitet sind, rückt mit DashPay nun das Hauptthema dieser Ausarbeitung in den Vordergrund. Wie schon in Kapitel 1 beschrieben, handelt es sich bei DashPay[7] um ein Data Contract, welches einer dezentralen Anwendung den Aufbau von direkten, bidirektionalen Zahlungskanälen zwischen den im Abschnitt 2.2 erwähnten Identities ermöglicht. Durch den im Abschnitt 2.3 behandelten Dash Platform Name Service ist es für Entitäten somit möglich, für DashPay einen Benutzernamen mit einer Identity zu verbinden. Dies ist eines der Hauptziele von DashPay und ein großer Fortschritt im Bereich der Nutzerfreundlichkeit von Kryptowährungen, da Nutzer nun nicht mehr gezwungen sind, Transaktionen durch Umwege (z. B. einscannen eines QR-Codes oder kopieren/abtippen) an eine Adresse zu schicken[37, 2]. Zahlungen werden außerdem dadurch vereinfacht, dass Benutzer nur einmalig ihre Kontaktdaten durch gegenseitige Kontaktanfragen austauschen müssen. Durch den eindeutig registrierten Benutzernamen ist jede teilnehmende Identity für alle anderen DashPay-Teilnehmer unverwechselbar zuordenbar. Alle nachfolgenden Zahlungen können anschließend durch die auf Drive hinterlegten gegenseitigen Kontaktanfragen und den darin enthaltenen Benutzernamen initiiert werden. Zusätzlich sind modifizierbare Benutzerprofile möglich, um z. B. Eigenschaften wie einen freiänderbaren Displaynamen, eine kurzgehaltene Biografie oder auch ein Avatarbild zu realisieren[37, 5] und somit ein Zahlungssystem mit der gleichen Benutzerfreundlichkeit wie z. B. PayPal anzubieten.

Dadurch das die öffentlichen Schlüssel, die den Zahlungsverkehr über das darunterliegende Dash Netzwerk ermöglichen, auf Drive verschlüsselt hinterlegt sind, können Drittparteien den durch DashPay koordinierten Zahlungsverkehr zwischen zwei DashPay-Nutzern nicht überwachen. Die durch den öffentlichen Schlüssel abgeleiteten Zahlungsadressen werden im Idealfall und zum Schutz der Privatsphäre nur einmalig verwendet[37, 5]. Da die beiden Zahlungsparteien jeweils ihren eigenen privaten Schlüssel und die gegenseitigen öffentlichen Schlüssel kennen, ist es jederzeit möglich, automatisiert eine vollständige und transparente Zahlungshistorie zwischen den Teilnehmern des Zahlungskanals zu rekonstruieren (ein Unterfangen, das in normalen Wallets unter der Nutzung von Adressen ohne Mehrfachverwendung, nicht ohne großen organisatorischen Aufwands seitens der Endnutzer möglich ist). In den folgenden Unterabschnitten werden die technischen Details des DashPay Data Contract detaillierter offengelegt, die notwendig sind, um dieses dezentralisierte Zahlungssystem überhaupt erst zu ermöglichen und die Funktionalität von DashPay in einem Dash Wallet zu integrieren. Außerdem wird auf den momentanen Stand des Projektes und das

derzeit ausgerollte Alpha-Programm[8], welches sich zum Zeitpunkt dieser Ausarbeitung (April. 2021) in der dritten von fünf Testphasen befindet, eingegangen.

### 3.1 DashPay Data Contract

Das Grundkonzept von DashPay basiert auf Kontaktanfragen, die sich die Benutzer des Zahlungsnetzwerks gegenseitig zuschicken können und den daraus entstehenden Beziehungen[37, 6] (im weiteren Verlauf der Ausarbeitung auch als *friendship*[37, 4] bezeichnet). Diese daraus resultierenden Beziehungen verbinden immer zwei Identities miteinander. Der DPNS für die Auflösung zwischen Identity ⇔ Name ist lediglich für die erleichterte Bedienung durch menschenlesbare Identifikationsmerkmale gegeben (der Bezeichner einer Identity ist 256 Bit lang, das merkt sich so schnell keiner, auch nicht in einer Base58-Darstellung[37, 4]), spielt aber auf technischer Ebene innerhalb einer Kontaktanfrage keine Rolle.

Um mit dem DashPay Data Contract zu interagieren, muss vom Benutzer sowohl eine Identity registriert werden als auch die Unterstützung von einen mit Identities umgehenden Hierarchical Deterministic Wallet[44] gegeben sein[37, 7]. Unterteilt ist das Data Contract in insgesamt drei unterschiedliche Dokumententypen, die in den folgenden Abschnitten einzeln behandelt werden.

- *contactRequest* → Kontaktanfragen (Aufbau von Beziehungen zwischen Identities + Schlüssel für den bei einer *friendship* zur Verfügung stehenden Zahlungskanal).
- *profile* → Benutzerprofile, um öffentlich einsehbare Informationen über eine Identity zu speichern.
- *contactInfo* → Kontaktinformationen, um private Informationen über andere Identities zu speichern.

### 3.2 Kontaktanfragen (*contactRequest*)

Das *contactRequest*-Dokument[37, 7.1] für die Realisierung von Kontaktanfragen beschreibt durch die enthaltenen Felder eine Einwegbeziehung zwischen einer absendenden und empfangenen Identity. In den Feldern `$ownerId` und `toUserId` werden die an der Kontaktanfrage beteiligten Identities persistiert. Klienten können durch diese beiden Felder in Kombination mit dem Feld `$createdAt` ihre eigenen gesendeten Kontaktanfragen als auch eingehende Kontaktanfragen von Drive heraus abrufen. Bei einer Neuinitialisierung werden erstmals alle vorhandenen Kontaktanfragen abgerufen, danach um Bandbreite zu sparen nur noch die, die den letzten Synchronisationszeitpunkt überschritten haben. Zwei Identities, die sich gegenseitig eine Kontaktanfrage geschickt haben, gelten durch diese Regeln automatisch als Kontakte. Um Kryptografie zwischen den beiden Identities (z. B. zur Geheimhaltung der gegenseitigen öffentlichen Schlüssel) zu gewährleisten, muss außerdem über den Diffie-Hellman-Schlüsselaustausch ein geteiltes Geheimnis (*shared secret*) generiert werden. Die Ableitung des *shared*

*secret* erfolgt durch Elliptic Curve Diffie-Hellman (ECDH) mithilfe der Bibliothek *libsecp256k1* [19]. Um durch ECDH einen Punkt auf der Kurve abzuleiten, existieren die Felder `senderKeyIndex` und `recipientKeyIndex`. Der Absender verwendet zum Ableiten seinen privaten Schlüssel an der Stelle `senderKeyIndex` und den öffentlichen Schlüssel an der Stelle `recipientKeyIndex`. Der Empfänger vertauscht die beiden Schlüssel und führt die gleiche Operation durch. Beide Ableitungen führen zum gleichen Punkt auf der ECDH-Kurve und somit zum gleichen *shared secret*.

Um die in BIP32[46] angesprochenen Mehrfach-Accounts in DashPay abzubilden, existiert das Feld `accountReference`. Identities repräsentieren wie in DIP13[44] beschrieben Entitäten und sind somit nicht accountbasiert. Die in DashPay verwendeten Kontaktanfragen verbinden also zwei Identities im Kontext von zwei Accounts miteinander. Innerhalb des `accountReference`-Feld wird also der Index des gewünschten Accounts zum Empfang von Zahlungen des Absenders einer Kontaktanfrage persistiert. Bei einer bereits bestehenden *friendship* sollten neue Kontaktanfragen einer bereits bekannten Identity mit einer anderen `accountReference` verworfen werden. Stattdessen sollte dem Nutzer beim Senden einer Zahlung an eine Entität eine Auswahl der bekannten Accounts angezeigt werden. Um in einem solchen Fall zwischen den verschiedenen Accounts zu unterscheiden, ist es möglich, durch das Feld `encryptedAccountLabel` einen für den Empfänger verschlüsselten Accountnamen mitzusenden. Der erweiterte öffentliche Schlüssel, den ein Empfänger einer Kontaktanfrage verwendet, um Zahlungsadressen für Zahlungen an den Absender abzuleiten, wird durch das Feld `encryptedPublicKey` repräsentiert.

Um Zahlungen auf verschiedenen Geräten mit unterschiedlich synchronisierten Blockhöhen zu erhalten, wird die Blockhöhe des zuletztbekannten Blockes mit ChainLock aus der Core Chain in dem Feld `$coreHeightCreatedAt` eingetragen. Klienten können diesen Wert durch die Platform Chain abfragen. Während der Validierung des Dokumentes werden nur Werte, welche in den letzten fünf bekannten Blockhöhen von Blöcken mit ChainLock liegen, akzeptiert. Falls nun zwei Geräte mit der gleichen Identity eine unterschiedliche Blockhöhe aufweisen und in der Zwischenzeit eine *friendship* mit einer anderen Identity gebildet wird und eine Zahlung von dieser stattfindet, kann das Gerät, welches von dieser Zahlung noch nichts mitbekommen hat, beim Eingang der Kontaktanfrage von dem darin enthaltenen Block aus neusynchronisieren und die Zahlung somit sichtbar machen.

Um Adressen für Zahlungen zu generieren, wird auf die in BIP32[46] besprochenen Ableitungspfade (*deviation path*), das in BIP43[36] besprochene Purpose-Feld und das in DIP9[41] erwähnte coin-spezifische Feld im Ableitungspfad zurückgegriffen. Für das Generieren eines nachweislich eindeutigen Ableitungspfads pro *friendship* für die Zahlungsadressen muss auf die in DIP14[42] vorgestellten 256-Bit Ableitungspfade zurückgegriffen werden. Da die in BIP32 definierten

Ableitungspfade durch die Limitierung auf 31-Bit maximal $2^{31}$ Adressen generieren können, stellt dies ein Problem bezüglich der Privatsphäre im Zahlungsverkehr dar. Der Ableitungspfad beinhaltet die Identities der beiden Benutzer, die deutlich länger als 31-Bit sind. Eine Reduzierung der Länge dieser hätte also eine Verringerung der Entropie zur Folge. Ein Angreifer könnte dementsprechend den gesamten Wertebereich von $2^{31}$ durchsuchen und mit den Werten die erweiterten öffentlichen Schlüssel, die es erlauben, den Zahlungsverkehr zwischen zwei Benutzern zu rekonstruieren, generieren. Um dies zu verhindern, werden stattdessen 256-Bit große Ableitungspfade verwendet, da ein Durchsuchen eines Wertebereichs von $2^{256}$ deutlich schwieriger ausfällt. Der von DashPay verwendete Ableitungspfad für die Generation des erweiterten öffentlichen Schlüssels ist in Abbildung 3 visualisiert. Dadurch das die letzten drei Felder veränderbar sind, kann mit diesem Ableitungspfad der gesamte Adressraum für alle möglichen Identities abgedeckt werden[1].

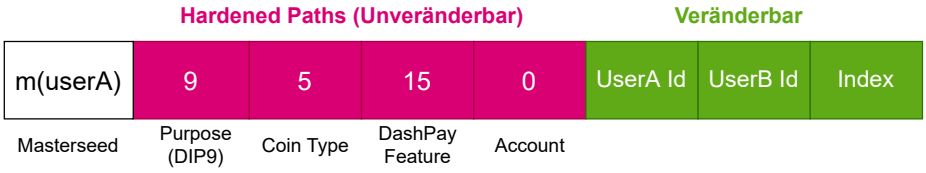


**Fig. 3.** Ableitungspfad, um einen erweiterten öffentlichen Schlüssel zu erstellen

*contactRequest*-Dokumente sind unveränderlich[2] und können nicht gelöscht werden. Dies ist eine Design-Entscheidung, um zu verhindern, das Benutzer ihre erweiterten öffentlichen Schlüssel ändern können, um zu verhindern, das der Ableitungspfad des vorherigen erweiterten Schlüssels verloren geht und es für die Nutzer so aussieht, als ob die vorherigen Transaktionen nie stattgefunden haben. Um eine Kontaktanfrage zu senden, müssen die in diesem Abschnitt gesamten Felder befüllt werden und das resultierende Dokument nach Signierung des Absenders im Netzwerk verteilt werden, um es für eine State Transtion auf der Platform Chain einzureichen.

In bestimmten Situationen kann es hilfreich sein, alle eingehenden Kontaktanfragen automatisch anzunehmen. Für diesen Spezialfall gibt es das Feld `autoAccept Proof`, durch das ein Schlüssel übergeben wird, der es erlaubt, die entgegengesetzte Kontaktanfrage selber zu signieren und einzureichen. Ein solcher Schlüssel könnte z. B. von einem Händler in einem QR-Code eingebettet werden. Dieser Schlüssel ist nicht ewig gültig und enthält durch einen Zeitstempel ein Verfallsdatum. Beispielhaft könnte eine automatische Kontaktanfrage in URI-Form so aussehen: `dash:?du=bobspizza&dapk=13SuoA8Z5tG9`.... Zu lesen: Sende eine

[1] Momentan sind mehrere Accounts durch DashPay noch nicht unterstützt. Deswegen ist das Account-Feld nicht modifizierbar.

[2] Ein Nachteil entsteht hier, falls ein falsch abgeleiteter erweiterter öffentlicher Schlüssel eingereicht wurde.

Kontaktanfrage an die Identity die im DPNS mit dem Namen *bobspizza* registriert ist, sende anschließend eine Kontaktanfrage von *bobspizza* aus an die eigene Identity und signiere diese mit dem Schlüssel aus dem `dapk`-Parameter.

### 3.3 Profile (*profile*)

Das *profile*-Dokument[37, 7.2] findet Verwendung, um öffentlich verfügbare Informationen über DashPay-Benutzer zu persistieren. Es wird durch das `$ownerId`-Feld einer Identity zugewiesen. Die weiteren möglichen Felder sind hier:

- `avatarUrl`: Optional. Eine URL die auf ein Avatarbild zeigt. Sollte nur für Kontakte heruntergeladen und gecached werden. Den externen URLs sollte nicht vertraut werden. Mit den optionalen Feldern `avatarHash` und `avatarFingerprint` kann zusätzlich die Authentizität des Avatarbildes sichergestellt werden.
- `publicMessage`: Optional. Eine öffentlich einsehbare Biographie, welche UTF-8 encodierte Zeichen enthalten kann und eine maximale Länge von 250 Zeichen aufweisen darf.
- `displayName`: Optional. Erlaubt es, einen persönlicheren Namen zu wählen als den durch DPNS registrierten DNS konformen Benutzernamen. Maximale Länge hier 25 Zeichen und UTF-8 ist ebenfalls unterstützt.
- `$createdAt` und `$updatedAt`: Zeitstempel in der Unixzeit um mit dem höchsten Zeitstempel immer die aktuellste Version des Profils zu finden.

Beim Veröffentlichen eines *profile*-Dokuments muss nur darauf geachtet werden, das alle Pflichtfelder ausgefüllt sind. Profile sind veränderbar und können jederzeit über State Transitions aktualisiert werden. In der Namenssuche können Profile nach dem Auflösen des eingegebenen Benutzernamens über DPNS per Identity nachgeladen werden.

### 3.4 Kontaktinformationen (*contactInfo*)

Über das *contactInfo*-Dokument[37, 7.3] können nicht-öffentlich einsehbare Informationen eines Benutzers bezüglich der bestehenden Kontakte gespeichert werden. Somit ist es möglich für Kontakte ein Alias oder eine Notiz festzulegen. Mit dem `displayHidden`-Feld ist es auch möglich, Kontakte auszublenden, was ein notwendiges Feature ist, da Kontaktanfragen unmodifizierbar sind.

### 3.5 Order of Synchronization

Da die Synchronisation sowohl die Layer 1 (Core Chain) und Layer 2 (Platform Chain) betrifft sollten Klienten DIP16 (*Headers First Synchronisation on Simple Payment Verification Wallets*)[43] implementieren[37, 9]. Anfragen an Dash Platform können erst stattfinden, nachdem die deterministische Liste der Masternodes geladen ist. Da in DashPay der erweiterten öffentlichen Schlüssel der

Kontakte in den Kontaktanfragen gespeichert sind, müssen erst alle Kontaktanfragen synchronisiert werden, bevor mit dem öffentlichen Schlüssel die Zahlungshistorie nachgebildet werden kann.[45] Um die Kontaktanfragen zu synchronisieren, muss der Zugriff auf die Liste der Masternodes bestehen. Um die Liste der Masternodes zu generieren, braucht ein Wallet die Blockheader vom Ende der Blockchain. Um dieses Synchronisationsproblem zu lösen, werden zuerst die fehlenden Blockheader angefragt um dann durch DIP4[34] die Liste der Masternodes zu generieren. Anschließend kann durch die DAPI per Drive die Liste der Kontaktanfragen heruntergeladen werden, um die Bloom-Filter zu konstruieren um die Zahlungshistorie per SPV zu rekonstruieren.

### 3.6 Aktueller Stand und zukünftige Pläne

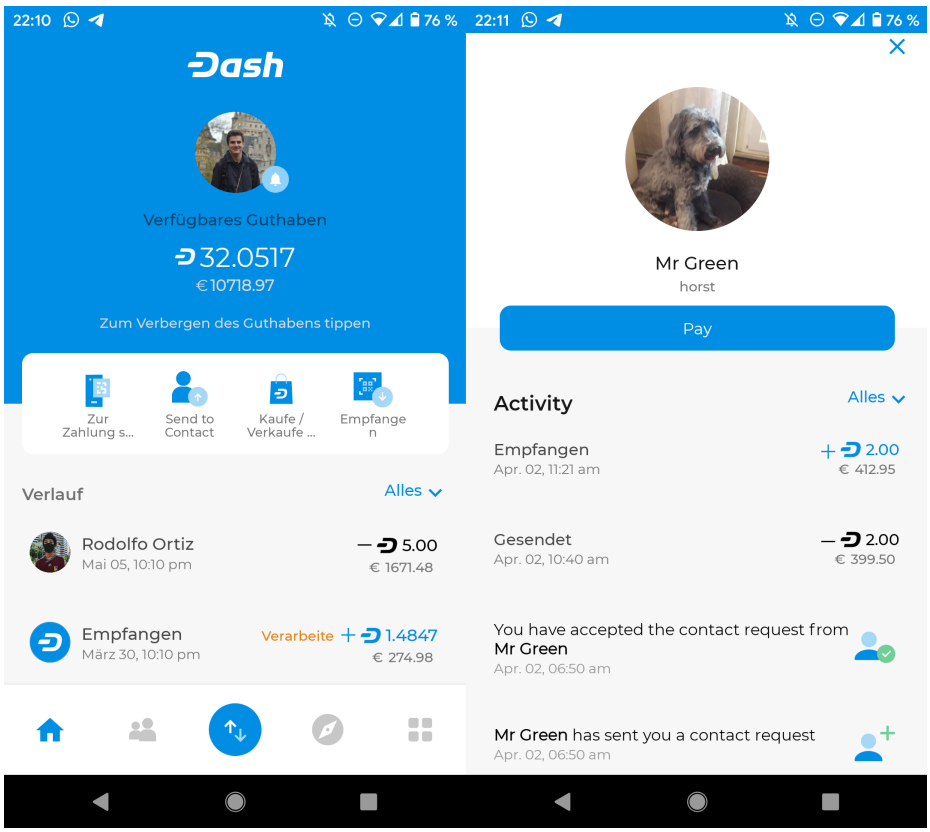


**Fig. 4.** DashPay Android-Referenzimplementierung des Alpha-Programmes (Version: *Dash Wallet 4.2-dashpay*)

Aktuell befindet sich DashPay in der vierten von fünf Runden des aktuell laufenden öffentlichen Alpha-Programmes[8]. In Abbildung 4 sind zwei Screenshots der mo-

mentanen im Testnetzwerk operierenden Android-Version abgebildet. Auf der linken Seite der Abbildung ist die Hauptansicht zu sehen, in der verschiedene Operationen wie z. B. das Hinzufügen von Kontakten oder auch tätigen von Zahlungen möglich ist. Im unteren Bereich befindet sich der gesamte Zahlungsverlauf, der sowohl Kontakte als auch Adressen beinhaltet. In der rechten Seite ist eine Detailansicht eines Kontaktes abgebildet. Der Nutzer mit dem DashPay Benutzernamen *horst* hat sich hier den Displaynamen *Mr Green* gegeben. Ein Profilbild ist ebenfalls zu sehen. In der Aktivität sind sowohl die Zahlungen als auch die gegenseitigen Kontaktanfragen dargestellt.

Die Benutzung von DashPay erweist sich schon in der Alpha-Phase als einfach und fehlerfrei. Die Interaktionen mit Dash Platform finden fast in Echtzeit statt und lässt durch die responsive Benutzeroberfläche kaum erahnen, das dahinter ein dezentralisiertes System steckt. Die Transaktionen auf Layer 1 werden über die Core Chain schnell und mit niedrigen Transaktionskosten abgewickelt. DashPay beweist als zweite dezentralisierte Anwendung nach DPNS, das Dash Platform mit dem Ansatz der Data Contracts einen nützlichen und zukünftig interessanten Technologie-Stack entwickelt. Aktuell sind auf der Roadmap für DashPay weitere Verbesserungen für die Benutzeroberfläche als auch Upgrades von Dash Platform auf zukünftige Versionen geplant. Da DIP15[37] am 12. Dezember 2020 erstellt wurde, ist hier eine rasche Realisierung von DashPay zu beobachten. Ein genauer Release-Termin für das Hauptnetzwerk ist nicht bekannt.

## 4 Fazit

DashPay beweist das auch Kryptowährungen von der Benutzerfreundlichkeit bekannter Zahlungsdienste wie z. B. PayPal oder CashApp profitieren können. Um eine Massenadaption von Kryptowährungen voranzutreiben, ist es zwingend notwendig, eine einfachere Methode zur Zahlungsabwicklung als dem bloßen kopieren von Adressen zu entwickeln. Mit DashPay besteht hier nun ein grundlegendes Konzept, das stetig verbessert werden kann. Als ebenfalls zuverlässig stellt sich die darunterliegende Dash Platform heraus, die das ganze erst ermöglicht. Der datenorientierte Ansatz der Data Contracts stellt sich als effektiv und zuverlässig zur Modellierung komplexer Sachverhalte heraus. Es bleibt abzusehen, inwieweit die Entwicklung von Dash Platform und DashPay in den folgenden Jahren voranschreitet und wann eine Überführung auf das Hauptnetzwerk stattfinden kann. Auch wird es interessant zu sehen, welche weiteren dezentralisierten Anwendungen auf der Dash Platform aufbauen werden. Ebenfalls bleibt abzusehen, wie DashPay zukünftig in Venezuela anklang finden wird. In dem von Hyperinflation geplagten Land nehmen schon seit den letzten Jahren vereinzelt Geschäfte die Kryptowährung Dash an[35].

## References

1. Cashapp, `https://cash.app/`

2. Concise binary object representation, https://cbor.io/
3. Dash, https://www.dash.org/
4. Dash core group, https://www.dash.org/de/dcg/
5. Dash sdk, https://github.com/dashevo/js-dash-sdk
6. Dashpay, https://www.dash.org/de/dashpay/
7. Dashpay, https://dashplatform.readme.io/docs/explanation-dashpay
8. Dashpay alpha-programm, https://www.dash.org/de/dashpay-alpha-programm/
9. Data contract, https://dashplatform.readme.io/docs/explanation-platform-protocol-data-contract
10. Decentralized api, https://dashplatform.readme.io/docs/explanation-dapi
11. Document, https://dashplatform.readme.io/docs/explanation-platform-protocol-document
12. Domain name system, https://de.wikipedia.org/wiki/Domain_Name_System
13. Drive, https://dashplatform.readme.io/docs/explanation-drive
14. Global cryptocurrency charts, https://coinmarketcap.com/de/charts/, abgerufen am 13.04.2021
15. Identity, https://dashplatform.readme.io/docs/explanation-identity
16. Infinite market cap, https://8marketcap.com/euro/, abgerufen am 13.04.2021
17. Introduction to smart contracts, https://ethereum.org/en/developers/docs/smart-contracts/
18. Json schema, https://json-schema.org/
19. libsecp256k1, https://github.com/bitcoin-core/secp256k1
20. Masternodes, https://www.dash.org/de/masternodes/
21. Name service (dpns), https://dashplatform.readme.io/docs/explanation-dpns
22. Paypal, https://paypal.com/
23. Paypal.me, https://www.paypal.com/paypalme/
24. Platform chain, https://dashplatform.readme.io/docs/explanation-drive-platform-chain
25. Platform consensus, https://dashplatform.readme.io/docs/explanation-platform-consensus
26. Platform protocol (dpp), https://dashplatform.readme.io/docs/explanation-platform-protocol
27. State transition, https://dashplatform.readme.io/docs/explanation-platform-protocol-state-transition
28. Stripe, https://stripe.com/
29. Testnets and devnets, https://docs.dash.org/en/stable/developers/testnet.html
30. Uncharted territory: why consumers are still wary about adopting cryptocurrency, https://www.kaspersky.com/blog/cryptocurrency-report-2019/
31. What is dash?, https://dashplatform.readme.io/docs/introduction-what-is-dash
32. What is dash platform?, https://dashplatform.readme.io/docs/introduction-what-is-dash-platform
33. Alibrandi, D.: An introduction to dash platform, dapi, and drive (2019), https://blog.dash.org/an-introduction-to-dash-platform-dapi-and-drive-9d080d6e89c9?gi=46c723272ee2, version vom 20.05.2019
34. Block, A., Westrich, S., Freer, A., UdjinM6: Simplified verification of deterministic masternode lists (2018), https://github.com/dashpay/dips/blob/master/dip-0004.md, version vom 16.03.2021

35. Nunziante, L.: Kryptowährung dash auf dem vormarsch in venezuela: Es tut sich etwas – zumindest in der mittelklasse (2018), https://www.gq-magazin.de/auto-technik/article/der-dash-auf-dem-vormarsch-in-venezuela-es-tut-sich-etwas-zumindest-in-der-mittelklasse
36. Palatinus, M., Rusnak, P.: Purpose field for deterministic wallets (2014), https://github.com/bitcoin/bips/blob/master/bip-0043.mediawiki, version vom 30.04.2019
37. Samuel Westrich, E.B.: Dashpay (2020), https://github.com/dashpay/dips/blob/master/dip-0015.md, version vom 16.12.2020
38. Shumkov, I.: Introducing the platform chain (2019), https://blog.dash.org/introducing-the-platform-chain-982fe6aea67f, version vom 17.10.2019
39. Shumkov, I., Suprunchuk, A.: Dash platform name service (2020), https://github.com/dashpay/dips/blob/master/dip-0012.md, version vom 16.03.2021
40. Shumkov, I., Suprunchuk, A., Westrich, S., Cofresi: Identities (2020), https://github.com/dashpay/dips/blob/master/dip-0011.md, version vom 16.03.2021
41. Westrich, S.: Feature derivation paths (2019), https://github.com/dashpay/dips/blob/master/dip-0009.md, version vom 16.03.2021
42. Westrich, S.: Extended key derivation using 256-bit unsigned integers (2020), https://github.com/dashpay/dips/blob/master/dip-0014.md, version vom 25.02.2021
43. Westrich, S.: Headers first synchronization on simple payment verification wallets (2020), https://github.com/dashpay/dips/blob/master/dip-0016.md, version vom 16.03.2021
44. Westrich, S.: Identities in hierarchical deterministic wallets (2020), https://github.com/dashpay/dips/blob/master/dip-0013.md, version vom 22.12.2020
45. Westrich, S.: The journey to dashpay (2020), https://blog.dash.org/the-journey-to-dashpay-3e604d17814c, version vom 23.12.2020
46. Wuille, P.: Hierarchical deterministic wallets (2012), https://github.com/dashevo/bips/blob/master/bip-0032.mediawiki, version vom 20.10.2020