Implementation and Evaluation of an Incentivized Blockchain-Based Deposit-Refund System

Bachelor Thesis

for the

Bachelor of Science

in Applied Computer Science

at Baden-Wuerttemberg Cooperative State University Stuttgart

by

Niklas Sauer

September 3rd, 2017

Time of Project	12 Weeks
Student ID, Class	2677254, STG-TINF15A
Company	Hewlett Packard Enterprise
Location	Böblingen, Germany
Supervisor	Ralph Beckmann
Reviewer	Wolfgang Weyand
Manager	Ricardo Fernandez Diaz

Declaration of Authorship

Hereby I solemnly declare:

1. that this Bachelor Thesis, titled *Implementation and Evaluation of an Incentivized Blockchain-Based Deposit-Refund System* is entirely the product of my own scholarly work, unless otherwise indicated in the text or references, or acknowledged below;
2. I have indicated the thoughts adopted directly or indirectly from other sources at the appropriate places within the document;
3. this Bachelor Thesis has not been submitted either in whole or part, for a degree at this or any other university or institution;
4. I have not published this Bachelor Thesis in the past;
5. the printed version is equivalent to the submitted electronic one.

I am aware that a dishonest declaration will entail legal consequences.

Stuttgart, September 3rd, 2017

Niklas Sauer

Preface

This thesis presumes background knowledge in the realm of blockchain technology. At the very least, the concepts presented by Bitcoin should be familiar to the reader in order to ensure full comprehensibility of the technical proceedings presented and comparisons drawn herein. Published by the National Institute of Standards and Technology, [50] may be consulted as a general introduction to this landscape. In concert, the author's knowledgeability has been demonstrated through the research conduced in [35].

Abstract

Contents

Acronyms VI

List of Figures VII

List of Tables VIII

Listings IX

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Goals and Scope . . . 2
- 1.3 Thesis Overview . . . 2

2 Theoretical Framework 3
- 2.1 Deposit-Refund System for Bottled Beverages in Germany . . . 3
  - 2.1.1 Legal Basis . . . 3
  - 2.1.2 Amendments . . . 4
  - 2.1.3 Operation . . . 6
- 2.2 Decentralized Applications . . . 10
  - 2.2.1 History & Raison d'Être . . . 10
  - 2.2.2 Definition . . . 11
  - 2.2.3 Tokens . . . 16
- 2.3 Ethereum . . . 18
  - 2.3.1 Overview . . . 18
  - 2.3.2 Developer Popularity & Adoption . . . 18
  - 2.3.3 Concepts . . . 19
  - 2.3.4 Functioning . . . 22

3 Concept 24
- 3.1 Solution Overview . . . 24
  - 3.1.1 Functional Requirements . . . 24
  - 3.1.2 Non-Functional Requirements . . . 24
- 3.2 Evaluation Framework . . . 24
- 3.3 Architecture . . . 24
  - 3.3.1 Assumptions . . . 24

4 Implementation 25

5 Evaluation 26

6 Conclusion and Discussion 27

7 Summary **28**

8 Outlook **29**

8.1 Enhancements and Additions 29

8.2 Adoption and Scalability 29

8.3 Additional Fields of Application 29

Bibliography **30**

Acronyms

dApp	Decentralized Application
DPG	Deutsche Pfandsystem GmbH
EAN	European Article Number
EOA	Externally Owned Account
EVM	Ethereum Virtual Machine
HPE	Hewlett Packard Enterprise
VerpackV	Verpackungsverordnung
VerpackV	Verpackungsverordnung

List of Figures

2.1 Aggregated quote of reusable beverage packaging (in %) between 1991 and 2013 [16, p. 1] 4
2.2 Criteria to be considered for deposits (valid until Jan 1st, 2019) [27, p. 9] . 6
2.3 Deposit-refund cycle [27, p. 14] 8
2.4 Clearing process for manual and automated bottle returns [27, p. 18] . . . 9

List of Tables

Listings

1 Introduction

1.1 Motivation

Compared to glass, plastic or aluminum packaging represents a lightweight and durable alternative. The impact of lightweight materials on shipping costs is non-negligible and has therefore been leveraged in the beverage industry for the past 30 years. Simultaneously, the quote of reusable bottles (Mehrwegflasche) has steadily fallen (from 72% in 1991 [16, p. 1] to 43% in 2015 [17, p. 4]), which prompted German lawmakers to introduce a system of returnable one-way bottles (Einwegflasche) in 2003 on which a deposit is paid [37, p. 53].

Contrary to expectations [27, p. 10], this regulation has not stopped the influx of one-way bottles but has rather benefitted bottlers. Whenever consumers pollute by leaving behind one-way bottles, an instant 25 cent profit – assuming that no one else has returned them – is generated for the producer. This passive profit was estimated to have reached up to 192M€ in 2011 alone [32, p. 245].

Ideally, this pollution of the environment should be punished by splitting non-claimed deposits of one-way bottles between environmental agencies and those consumers who regularly purchase reusable bottles, which save more resources. As a further consequence, those consumers who repeatedly neglect to return their one-ways should be required to pay a higher deposit. Such a revised approach can hopefully maximize the number of returned one-way bottles and effectively steer users towards reusable ones. Otherwise, a further decline may be inevitable and is shown to have had a direct negative impact on global warming, in addition to the excess amount of waste produced hereof [2].

When considered on a case-by-case basis, this problem inherently deals with deposits of very low extrinsic value. Moreover, such a system has to manage account balances and track the movement of value, increasing the appeal and likelihood for external attacks. Therefore, implementing the proposed approach by utilizing smart-contracts and micro-transactions on the Blockchain may come to mind upon choosing the underlying architecture. But how does such an implementation look like; are there special considerations to be made?

1.2 Goals and Scope

Since no research on the feasibility of this approach has been undertaken for an application as specific as this one, the study focuses on the end-to-end development process, including design, implementation and deployment. The objective of the research is to guide Hewlett Packard Enterprise (HPE) and interested parties alike in developing a mindset suitable for migrating applications onto decentralized platforms and evaluate if this Blockchain-based implementation can withstand the requirements of an incentivized deposit-refund system. The results are relevant as they reduce the go-to-market time of the previously outlined solution to stop the influx of one-way bottles, which in turn reduces pollution and warming of the earth (comp. 1.1), a serious concern to today's society. Moreover, HPE can offer more profound Blockchain-related consulting services through the insights gained.

It shall be explicitly noted that the study does not cover analyzing the effectiveness of employing such a system (i.e. reducing the usage of one-way bottles) or verifying wether this represents the best possible approach. Similarly, the legal framework and ethical questions pertaining to its usage are disregarded.

1.3 Thesis Overview

Chapter 2 reviews the relevant literature and is intended to answer all descriptive research questions that help define the project variables. The key concepts are then applied in chapter 3 as part of the architectural overview derived from the functional requirements of an incentivized deposit-refund system given earlier in the chapter. At the same time, the criteria to evaluate the forthcoming implementation are selected. Chapter 4 then describes the procedure to implement the solution, after which the actual evaluation may take place in chapter 5. Chapter 6 uses the results to draw a conclusion, discuss probable alternatives, as well as any limitations encountered during the study. Finally, chapter 7 summarizes the thesis's goals, methodology and results and is followed by chapter 8 to highlight interesting related research opportunities.

2 Theoretical Framework

2.1 Deposit-Refund System for Bottled Beverages in Germany

2.1.1 Legal Basis

Anticipating the depletion of capacity in disposal sites and due to the considerable share in waste generated by packaging [1] [27, p. 2], German federal government enacted an ordinance regarding the avoidance of packaging waste (*Verordnung über die Vermeidung von Verpackungsabfällen*, short: *Verpackungsverordnung (VerpackV)*) in 1991 which stipulates that packaging [43, § 1]:

- must be minimised to an extent necessary for protection and marketing of goods
- must be reused where possible
- must be recycled if reuse is not applicable

These objectives were supported by introducing the:

Obligation to take back packaging

Producers and distributors of packaging are obliged to take back packaging free of charge, restricted to those goods of type, shape, size and material found within their stock [43, §§ 4-6]. Distributors with a retail area of less than $200m^2$ are further exempted to the same brands. This duty may be only be ignored if a distributor participates in a system which ensures the periodical collection of waste [43, § 6], known and implemented as a refuse recycling system (*duales System*) in 1990 [27, p. 3].

Obligation to levy deposits on beverage packaging [2]

A deposit is to be charged by the distributor that will be refunded to the purchaser upon return of the bottle. This duty is applicable on all levels of trade involving domestic beverages sold in non-reusable packaging (*one-way packaging*) [43, § 7] and

[1] Recycling rate of packaging was below 50% in the early 1990s (20% for aluminium and plastic) [27, p. 2].

[2] Information regarding the assumed effects and critique thereof can be found in [45, p. 630] and [44].

becomes effective as soon as the quote of reusable beverage packaging falls below 72% [1] [43, § 9].

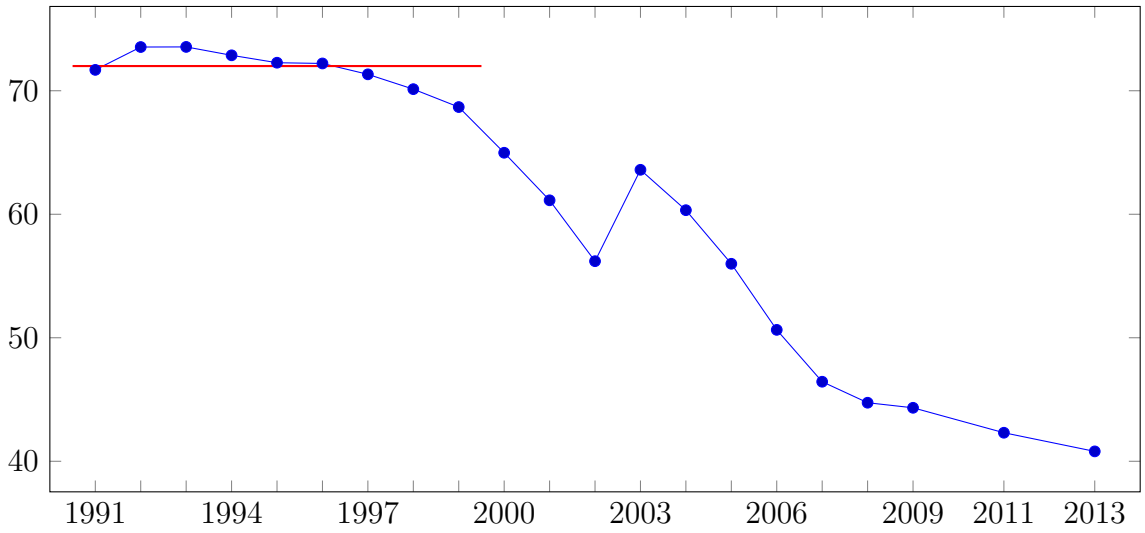

Figure 2.1: Aggregated quote of reusable beverage packaging (in %) between 1991 and 2013 [16, p. 1]

Starting in 1997, the threshold to levy deposits has been surpassed steadily (comp. Figure 2.1), requiring an additional assessment of the situation to ensure that the fluctuation does not represent a short-term development [43, § 9] [27, p. 5]. Although obvious at glance, the outcome of a compulsory deposit-refund system only became official on July 2nd, 2002 [37, p. 49], after a lawsuit lead by multiple bottlers and distributors had delayed the initial announcement [21]. Introduction of this mandatory system was scheduled for Jan 1st, 2003 [37, p. 53].

2.1.2 Amendments

The German packaging ordinance underwent several revisions, including a change of title to highlight the importance of recycling [42]. In the following, the most important changes affecting the current state (see Figure 2.2) shall be highlighted.

1998

Trims deposit obligation to those beverages for which the quote of reusable packaging has fallen when compared to 1991, though still necessitates that overall aggregated quote undercuts threshold of 72% [23, pp. 142]. Even though all five segments (beer, mineral water, carbonated soft drinks, fruit juice/non-carbonated soft drinks and wine) have failed this comparison [16, p. 1], juices/non-carbonated soft drinks and

[1] Aggregated quote derived from weighted average of percentages of reusable packaging encountered across individual segments in 1990 [43, § 9] [34, p. 134]

wine are exempted because their decline and market volume has not been regarded significant enough to justify the costs introduced with such a system [27, pp. 6, 9].

figure / table showcasing development of reusable packing quote in each segment (appendix)

2005

Reduces different deposit classifications to single deposit worth 0.25 € valid for all applicable beverages with a filling volume between 0.1 - 3.0 litres. Furthermore, ecologically advantageous packaging is admitted the same treatment and classification as that of reusable packaging which represents an important shift of thought since the sole utilisation of packaging has been considered inferior to its reuse with regard to all ecological aspects [27, p. 7]. Accordingly, the new goal is to promote the use of ecologically advantageous packaging with a target of 80% market dominance [25, § 1]. This amendment also adds non-carbonated soft drinks and mixed alcoholic drinks to the list of beverages subject to deposits (irrespective of the reusable packaging quote experienced) while simultaneously protecting dietary products from deposits [22, p. 1408] [24, p. 171]. Finally, retailers are required to take back any bottles made from a beverage packaging material (glas, metal, paper and plastic) also sold by that store (brand equality for retail areas of less than $200m^2$ still applies) [27, p. 11]. Previously, return had been limited to bottles of same shape, size [..] and type (comp. 2.1.1). This regulation attempts to stop isolated deposit-refund systems which arose because discounters created their own specially shaped bottles [24, p. 168]. To conform with this change, industry and commerce established the Deutsche Pfandsystem GmbH, responsible for setting up a nationwide deposit-refund system [27, p. 8].

2008

Orders distributors to clearly mark bottles as being obliged to a deposit. Further, distributors are demanded to participate in a nationwide deposit-refund system to enable the mutual settlement of claimed deposits [25, § 9]. Lastly, the exemption for dietary products is reduced to infant nutrition. This acts as a measure to counteract increasingly false product declarations attempting to forego deposits [25, pp. 531] [24, p. 171].

add 2019 amendment

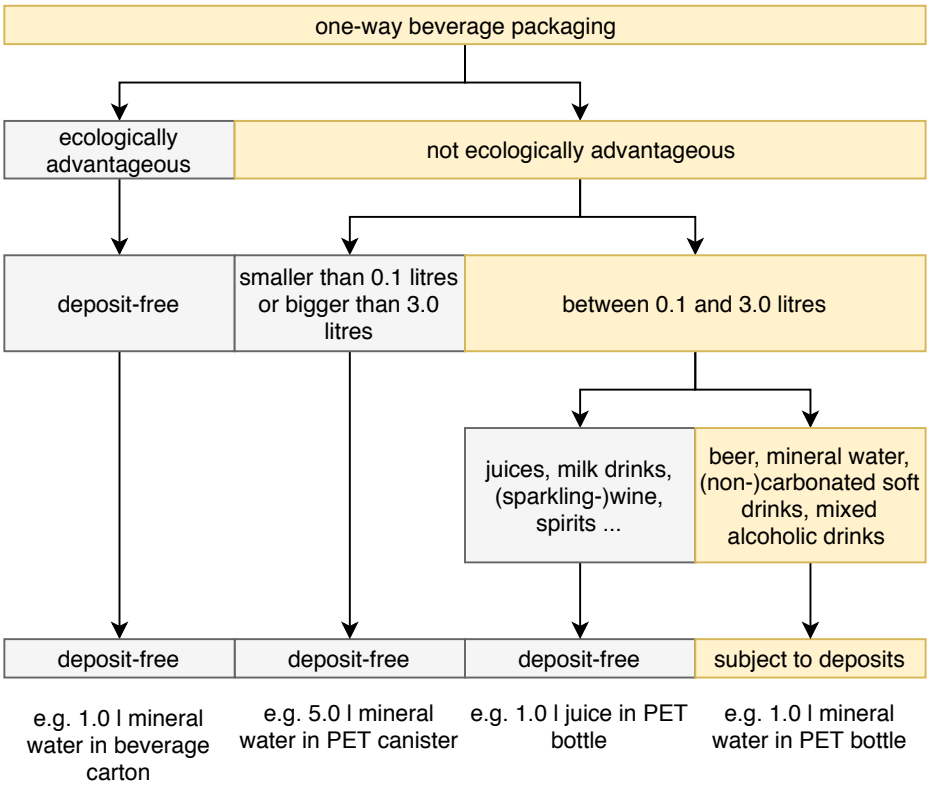

Figure 2.2: Criteria to be considered for deposits (valid until Jan 1st, 2019) [27, p. 9]

2.1.3 Operation

Administration by Deutsche Pfandsystem GmbH

VerpackV simply assumes that a nationwide deposit-refund system will be installed by May 1st, 2006 without further specifying its exact implementation details [22, Art. 2] [25, § 9]. Therefore, Deutsche Pfandsystem GmbH (DPG) has been founded in 2005 to ensure the definition of a standards framework, which includes IT-processes, APIs and a universal security mark identifying one-way packaged bottles on which a deposit is paid (*returnable bottle*). Other tasks concern the management of contracts, certifications and maintenance of a centralised database which stores information about products, participants and reverse vending machines [27, pp. 13]. However, DPG does not have access to purchasing data (e.g. number of sold and returned bottles or total amount of reimbursed deposits). This data is only exchanged between actors of the deposit-refund cycle. Consequently, DPG is not involved in the process of settling refund claims between take-back points and does not act as a central clearing house. It is exclusively responsible for providing a contractual and administrative basis needed to enable such a system [27, p. 14].

figure showcasing universal security mark (appendix)

Roles

As indicated, actors of the deposit-refund cycle (e.g. retailer selling returnable bottles) must register with DPG, accept the contractual basis for the role to be exercised and pay a yearly membership fee in order to participate. The most important roles are [27, pp. 15-16]:

Initial distributor
: Puts returnable bottle into circulation. Inherently tied to role of deposit escrow account manager. Must pay a one-time fee to register a beverage (i.e. an EAN) with DPG's central database. This fee depends on the number of pieces produced within a year for that beverage.

Deposit escrow account manager
: Administrates and disburses money received from levying deposit on returnable bottle. Task may be delegated to a service provider.

Take-back point
: Refunds consumer in exchange for returning bottles subject to deposits. Inherently tied to role of refund claimant.

Refund claimant
: Puts in a claim to be reimbursed for making advance refunds. Task may be delegated to a service provider.

Counting center operator
: Takes over role of reverse vending machine by confirming number of genuinely accepted bottles with a receipt.

Deposit-Refund Cycle

Each buyer of a returnable bottle must directly pay the designated deposit to the seller. This process starts with the initial distributor (step 1 of Figure 2.3) and is repeated for each sub-distributor until the bottles reaches retail which acts as the final distributor to end consumers (step 2 of Figure 2.3). After consumption, the consumer may return the packaging at any retail store obliged to take back the packaging in question (take-back point) (step 3 of Figure 2.3). Return may be performed manually (step 1a of Figure 2.4) or can be automated through reverse vending machines (step 1b of Figure 2.4). The former requires personnel to count the number of returned bottles and refund the consumer accordingly. Following count, the collected bottles must be shipped to a counting center which is responsible for digitally capturing each bottle (step 2a of Figure 2.4). At last, the

bottles will be compressed to eliminate the possibility of repeated refunds. In the case of reverse vending machines, the process of counting, capturing and compressing bottles can take place locally at a retail store (step 2b of Figure 2.4). Alternatively, retailers are permitted to return the packaging to the previous distributor, thereby unwinding the deposit-refund chain. However, this practice is rarely exercised since the packaging will not be reused and transportation would only induce additional ecological and financial strain. Instead, the compressed packaging can be sold directly to recycling companies (step 3 of Figure 2.4) [27, p. 16-17] [3].

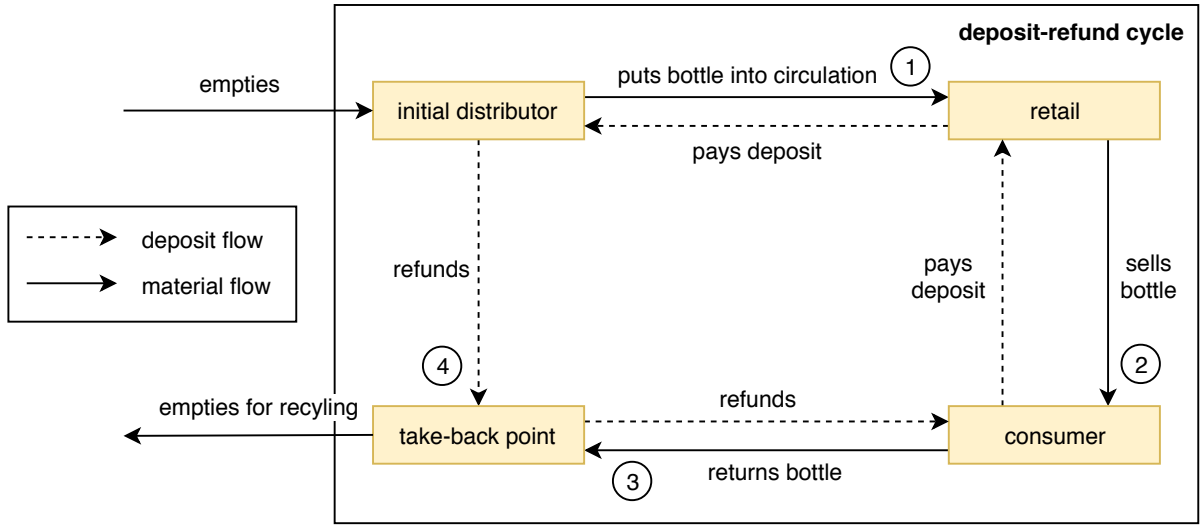

Figure 2.3: Deposit-refund cycle [27, p. 14]

Since take-back points refund consumers independent of a bottle's purchasing location, a settlement process (*clearing*) is required to reimburse retailers for making advance refunds (step 4 of Figure 2.3). This clearing happens by issuing a refund invoice to the initial distributor who can use the deposits originally collected the pay the bill (step 4 of Figure 2.4). Ideally, all accounts are in balance, meaning that all packaging has been returned. Otherwise, a surplus on the part of an initial distributor is to be experienced [27, p. 17].

Refund invoices must also include data representing the number of bottles for which a refund was issued. This data originates from the return process in which each bottle is eventually captured. Capture refers to both validating the universal security mark printed on a bottle's label as well as extracting the European Article Number (EAN) by scanning its barcode. By outfitting reverse vending machines with a network connection, looking up the EAN in question becomes possible, thus ensuring that a deposit for the bottle has been collected beforehand. For each valid bottle, a signed record is then produced and stored on the machine. This data must be retrieved and transmitted within one week. Likewise, retailers receive a similar dataset when relying on a counting center. Finally,

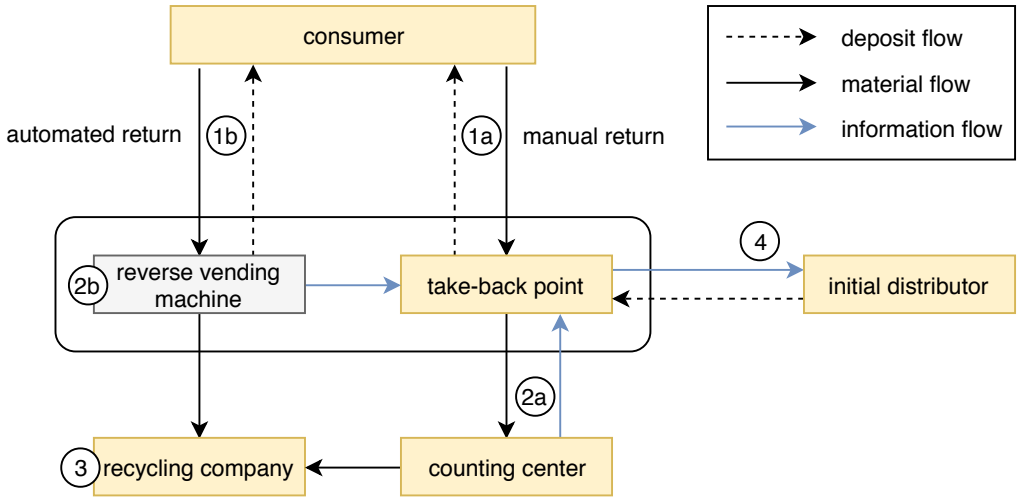

Figure 2.4: Clearing process for manual and automated bottle returns [27, p. 18]

any transmitted data is anonymised by removing the return location and summing up the refunds for each EAN captured. Initial distributors must pay the invoice within ten business days or five-teen calendar days at the lastet [27, p. 18-19].

explain intention / presumed effects of bottle deposits?

explain situation surrounding deposits on reusable bottles?

2.2 Decentralized Applications

2.2.1 History & Raison d'Être

Johnston's Law states that everything that can be decentralised, will be decentralised [1]. Fittingly, a new model for building massively scalable, successful and profitable applications, known as *Decentralized Applications (dApps)*, is emerging [33, p. 5] [28, pp. 1-2], suggesting that this will also become the fate of the vast majority of web software applications as those follow a centralized server-client model in which individuals directly depend on a central power to send and receive information [33, pp. 7-8].

The architectural structure embodied by the Web has not always been as pronounced as is these days. In its early days, people would host personal servers for others to connect to and everyone owned their data. But soon, it was recognised that one individual or group could pay for the maintenance of a server and profit from users that utilize it (e.g. Amazon Web Services [1]) and since this was easier to the average user, both conceptually and programmatically, the transition to few centralized controlling entities started [33, pp. 14].

Now, users willingly give their data to service providers in return for free usage, trusting them to not misuse or sell the data elsewhere [33, p. 24]. However, Snowden proved that this trust can, has and will be broken as long as we entrust (encrypted) data to central entities which represent a surveillance state's dream [14] [33, p. 25]. In addition, centralized infrastructure and services increase the chances for downtime, censorship and counterparty risk [13, p. 23]. Drawbacks like these have attributed to the development of dApps, first fully realized by Bitcoin [2] as a currency [28, p. 1] [30, p. 1]. Simultaneously, recent apps (e.g. Uber and AirBnb) attempt to decentralize real world parts of a business by providing a central and trusted data store, foreshadowing the development of decentralized apps beyond pure finances [3] [33, p. 15].

The concept of dApps is meant to take the web to its next [4] natural evolution [13, pp. 34]. Web3 represents the vision of a serverless [5] internet, encouraging applications to incorporate decentralised protocols in order to put users in control of their own data, identity and destiny [5]. As global economy evolves, data will become the primary form of value [33, p. 25], leading Raval to believe that:

[1] Amazon controls roughly 40% of the cloud market and serves customers like AirBnb, Expedia, Netflix, Slack and Spotify [15]. Please install [4], to experience the impact of an AWS downtime.
[2] BitTorrent, as an earlier example, depends on centralized trackers for data discovery [33, p. 26].
[3] Supports the blockchain evolution theorem proposed in [39].
[4] Web 2.0 describes the evolution towards user-generated content, responsive interfaces and interactivity [13, p. 34].
[5] Not to be confused with the centralized Function-as-a-Service offerings detailed in [36].

> "[dApps] will someday become more widely used than the world's most popular web apps." [33, p. 5]

2.2.2 Definition

Developers have different opinions on what exactly a dApp is or which components constitute it. Some think that no central point of failure is all it takes, whereas others name more specific requirements [33, p. 9]. The following will chronologically examine the various definitions available in order to derive one used for the remainder of this thesis.

2014

Buterin introduces the concept of a dApp as part of Ethereum's white paper. A complete dApp should consist of:

- low-level business-logic components
- high-level graphical user interface components

He goes on to state that "the blockchain and other decentralized protocols will serve as a complete replacement for the server for the purpose of handling user-initiated requests. ... Decentralized protocols ... may be used to store the web pages themselves." [19, p. 34].

This vague explanation is expanded through a consequent blog post (2014) in which Buterin formally defines the concept as being a superset of *smart-contracts* [18]:

Smart-contract
: Mechanism involving digital assets and a fixed number of parties, where some or all of the parties put assets in and assets are automatically redistributed among those parties according to a formula based on certain data that is not known at the time the contract is initiated.

Decentralized application
: Similar to a smart-contract but with unbounded number of participants and not necessarily of financial nature.

2015

To be considered a dApp, Johnston et al. assume that an application [28, p. 2]:

- is **open-source**, operates autonomously and adapts its protocol with user consent
- stores its data and records of operation in a **public blockchain**
- uses a **token** necessary for accessing it and rewarding contribution of value
- generates its tokens according to a predefined **cryptographic algorithm**

Moreover, he classifies dApps based on whether they have their own blockchain (Type I), use that of another dApp (Type II) or use a Type II dApp as its underlying protocol (Type III). The latter is also known as a protocol [28, pp. 3-4].

2016

Although similar to the elements proposed priorly, the definition given in *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology* highlights their importance and by such requires that a profitable dApp [33, pp. 9-14]:

- is **open-source** to achieve transparency and gain trust among users
- achieves **decentralized consensus** on application-level constructs (e.g. high-score)
- uses an **internal currency** for access, reward and monetisation
- has **no central point of failure** so that it cannot be shut down

2018

Finally, Antonopoulos and Wood (co-founder of Ethereum) back up and clarify Buterin's design guidelines by stating that a dApp is composed of at least [13, p. 34]:

- **smart-contracts** on a **blockchain**
- a web frontend **user interface**

Optionally, a dApp may rely on other decentralized components such as:

- decentralized storage
- decentralized messaging

explain that classification has been slimmed down to Type I & II dApps

Derived Definition

An application is considered to be a Decentralized Application (dApp) if it exhibits all of the characteristic features marked as a *must*. Secondary or recommended traits are labelled as *can* or *should* respectively.

Open-source

Traditional business models require the service for sale to be better than that of a competitor. By open-sourcing a solution (both frontend and backend), copying becomes inevitable [33, p. 10]. However, users will want to remain with the team that is best suited to maintain the application [33, p. 11]. Therefore, developers should not fear trading secrecy for transparency. Transparency removes a barrier to adoption as users are no longer required to trust that an application works as advertised [33, p. 9]. Additionally, open-source development often sparks the attraction of enthusiastic developers who contribute freely [33, p. 11].

- *must* be deployed visibly (i.e. the executed code can be inspected by a user)
- *should* be developed on an open-source platform

Johnston et al. call for an application to only alter its protocol by the consensus of its users. Yet, it can be argued that this does not represent a core property of a dApp since users can revert to the desired state at any time by forking the project.

- *can* require the consensus of users to upgrade its protocol

Decentralized consensus

Usually, application-level constructs (e.g. usernames, high-scores or status updates) are managed and modified only through the centralized application provider, thereby ensuring a single and consistent state (*singleton*). Luckily, blockchains provide the means to reach consensus in a decentralized manner. Traditionally, this is only applied to transactional logs [30, p. 1]. Extending this consensus mechanism to cover the outcome of a computer program and consequently storing the result on a blockchain for future retrieval as an input, would enable virtually any application. Seeing that such an application would always have a single globally valid state, the underlying infrastructure can be colloquially termed a world-computer.

Raval, Antonopoulos and Wood, refer to these programs as smart-contracts and equally expect them to be secured by and executed on a blockchain [33, p. 13] [13,

pp. 23, 34]. It must be noted that Buterin consistently denotes smart-contracts as being a subset of contracts. The former is of financial nature and may involve redistributing digital assets among parties or unlocking value depending on the conditions met [19, pp. 1, 13]. The latter can be used to encode arbitrary state transition functions required to build decentralized applications which may also issue tokens [19, pp. 1, 19]. For simplification purposes, the business logic of both applications types will be referred to as smart-contracts.

- Type I *must* achieve decentralized consensus on state-transitions
- Type II *can* only modify its data via a smart-contract

Finally, in order to achieve decentralized consensus, it is ultimately necessary to have a complete picture of the state-transitions (*transactions*) which have been applied by a network. While theoretically possible, applications have never been observed to only store their transactions, due to the long processing needed to rebuild the most current state (*data*).

- *must* store its transactions in a public blockchain
- *should* store its data in a public blockchain

Internal currency

According to Raval, profit is a cornerstone of any successful, robust and sustainable application [33, p. 9]. Because traditional revenue streams (e.g. fees, advertising, subscriptions) are not applicable without preventing the possibility of a fork, he proposes an internal currency [1] that is required to use the service [33, p. 11], hoping that it will appreciate in value by being listed on an exchange. Nevertheless, not every application is designed for profit.

- *can* use an internal currency for monetisation
- *can* use an internal currency for access / service usage

On the other hand, using an internal currency to reward the contribution of value is especially important for Type I dApps which employ their own blockchain and thus require an incentivization structure that keeps both miners and developers active [33, p. 9].

- Type I *must* use an internal currency to reward the contribution of value
- Type II *can* use an internal currency to reward the contribution of value

[1] Type I: Blockchain-native cryptocurrency, Type II: Smart-contract-issued token.

Lastly, Johnston et al. specify that tokens shall only be generated according to a predefined cryptographic algorithm. This is inherently true for Type I dApps which issue their internal currency as a reward for completing the cryptographic challenge (e.g. proof-of-work) [30, pp. 3-4]. In this context, tokens are considered to be a feature of Type II dApps (comp. Decentralized consensus). Therefore, their generation is not directly tied to cryptographic algorithms, but rather to the program's control flow.

- Type I *must* use a cryptographic algorithm to generate its internal currency
- Type II *can* only generate its internal currency via a smart-contract

Decentralized protocols

Although often required, storing massive amounts of data in a blockchain defats its design purpose of a simple transactional log. Moreover, this would diminish the incentive to maintain the network because the cost to participate (i.e. disk storage and bandwidth) will become considerably higher than the reward paid if many applications follow this trend. Hence, other solutions should provide methods for storing data in a decentralized way that is robust and as trust-less as possible [33, pp. 25].

- *can* use a decentralized storage solution

Similarly, message-intensive applications should not (entirely) rely on a blockchain for operation. This is due to the fact that every message will be stored forever and is public by default.

- *can* use a decentralized messaging solution

The definition presented within this section purposely does not make any assumptions about the user interface or a dApp's resiliency. More importantly, the primary focus and goal of any dApp developer should be to give users the confidence that a product works as promised by solely relying on open, decentralized peer-to-peer infrastructure services that can be inspected for functioning from end-to-end.

draw comparison to CLIs in respect to UI absence

compare infrastructure model against traditional web apps to highlight open deployment

2.2.3 Tokens

By introducing a digital equivalent, blockchain has evoked a semantic change for the word token originally used to mean privately-issued coin-like items that have insignificant value and are designed for a single purpose (e.g. transportation-, arcade- or laundry tokens). Now, tokens are used to simultaneously convey ownership, access rights, as well as the right to vote, for instance. Moreover, they are more easily exchangeable, a pitfall commonly cited [13, p. 173]. According to Antonopoulos and Wood, this could make tokens quite valuable when used in a way that does not re-create the conditions that made physical tokens worthless [13, p. 178], hereby hoping for industry wide solutions.

Use Cases

Antonopoulos and Wood present the following list of possible uses. However, tokens often encompasses several of these features and, much like their physical equivalents, it is hard to discern between them (e.g. driving license as identity and attestation) [13, pp. 173-174].

- Currency
- Resource
- Asset
- Access
- Equity
- Voting
- Collectible
- Attestation

Further, most projects employ tokens for two reasons: *utility* and/or *equity*. Similar to access tokens, utility tokens are those where the use of the token is required to pay for a service, application or resource [13, p. 176]. Yet, this requirement increases the barriers to adoption as the risks of broader economy (e.g. exchanges, liquidity, regulators) are added to that of the underlying blockchain technology [13, pp. 177]. Tokens should be adopted because the application cannot work without it, not because it presents a fast way to raise money while being disguised as a utility token to forego public securities regulations [13, p. 178].

Fungibility

Tokens are fungible if any single unit can be substituted for another without incurring a difference in value or function. A token is not entirely fungible if its historical provenance can be tracked, as this enables black- and whitelisting. The latter is the case for most

cryptocurrencies [1]. Lastly, non-fungible tokens represent unique tangible or intangible items which are not interchangeable [13, p. 175].

Intrinsicality & Counterparty Risk

While some tokens represent blockchain-native digital items, many tokens are used to represent extrinsic things. By such, consensus rules do not entirely apply and human law or agreements decide whether the transfer of ownership through tokens will be recognised. It is therefore even more important to understand who possess the associated asset and know what formal rules apply [13, pp. 175-176]. Leveraging tokens and smart-contracts to eliminate counterparty risk will most likely play part in fut ure market strategies.

[1] Monero pursues complete anonymity and therefore faces the possibility to be banned in Japan [12].

2.3 Ethereum

2.3.1 Overview

In line with the web3's vision of a serverless internet (comp. 2.2.1), Ethereum was designed to abstract the details of a blockchain to provide a deterministic and secure programming environment for decentralized applications in which developers are no longer required to bootstrap the underlying mechanisms [13, p. 27]. By such, it is practically regarded as an open-source, globally decentralized computing infrastructure that executes programs, known as smart-contracts, for which the current state and transitions leading to this state are stored in a blockchain alongside a cryptocurrency called Ether. However, it must be emphasised that Ether is primarily intended as a utility currency [1] to meter and constrain resource usage during execution which stands in stark contrast with Bitcoin's sole purpose of being a digital payment network [13, p. 23] [30, p. 1].

Compared to a general purpose computer, the main differences are that [13, p. 28]:

- state changes are governed by rules of consensus
- state is distributed globally on a shared ledger

From a computer science perspective, Ethereum may therefore be described as a deterministic but practically unbounded state-machine characterised by its [13, p. 23]:

- globally accessible singleton state
- virtual machine that applies changes to that state

2.3.2 Developer Popularity & Adoption

Naturally, and as is the case with most base layer technologies, Ethereum does not represent the only smart-contract platform available [2]. Nevertheless, several key figures support the claim of Ethereum dominating this space:

- 94 of top 100 tokens (i.e. dApps) launched on Ethereum (87% of top 800) [41] [29]
- ranks well above competitors (#33 vs. #72 for EOS) when comparing traffic of dedicated sites in StackExchange network [6]

[1] Amusingly, Vitalik Buterin, co-founder of Ethereum, would like to use Bitcoin Cash for day-to-day transactions [48].

[2] Contenders include (alphabetically listed): Cardano, Corda, EOS, Hyperledger Fabric, Lisk, NEM, NEO, Stratis and Waves.

Moreover, Wang and Vergne argue that upon emergence of a new, technically more advanced platform, the blockchain with a stronger development team and open-source community is more likely to succeed [46, p. 14]. This developer and community focused nature is clearly seen within the Ethereum landscape:

- StackExchange topic encounters mainly developer focused questions [40, p. 6]
- ~14.3k answered questions (vs. ~850 for EOS) [68% vs. 81% of total questions] [6]
- ~17.5k repositories on GitHub (vs. ~3.4k for EOS & ~2.5k for Hyperledger) [1]

Most importantly, Ethereum is estimated to have around 250k developers [29], a 30-fold lead to Hyperledger Fabric, the second most active community, according to a report from Gartner [20].

Similarly, the degree of decentralization should not be neglected when considering platforms based on public permissionless blockchains. To this extent, Ethereum boosts ~17.6k nodes across six continents [2] [9].

2.3.3 Concepts

In order to provide developers with a general-purpose computing architecture that runs programs everywhere, yet produces a common state secured by the rules of consensus [13, p. 31], Ethereum redefines existing concepts, while also introducing several new ones.

Accounts

At its core, the state of Ethereum is composed by objects called *accounts*, each of which has its own 20-byte address and features the following properties [47, p. 17]:

- Nonce
- Current ether balance
- Contract code (if present)
- Storage (empty by default)

Further, two types of accounts are distinguished [47, p. 17]:

[1] A fuzzy, top-level search has been conducted on GitHub on August 15th, 2018.

[2] EOS does not qualify since the 21 block producers used in its delegated proof-of-stake consensus mechanism are elected from a list of pre-approved candidates [26, p. 6] [7]. A similar case can be made for NEO which is only currently in the process of decentralization [31] [8].

Externally Owned Accounts (EOAs)

Accounts created by or for human users of the Ethereum network. This implies the existence of a private key which controls access to the associated funds [13, p. 13]. Such an account has no code, but can send messages by creating and signing a transaction.

Contract accounts

Accounts containing code that executes whenever a message is received from another account [13, p. 13]. Consequently, they are owned and controlled by the logic of the code stored respectively [13, p. 57]. This code allows contract accounts to read and write to internal storage and send messages. The latter unlocks additional capabilities such as transferring Ether and creating new or interacting with existing contracts [47, p. 19].

Messages & Transactions

As previously indicated, a transaction is a signed data package that stores a message which originated from an EOA. It works as expected from any cryptocurrency (i.e. it results in a transfer of value, if confirmed and non-zero, positive amount was specified) but has been expanded by three additional fields for contract invocation [47, p. 18]:

- recipient
- signature (identifies sender)
- amount of ether (denoted in Wei [1])
- *data* (optional)
- *start gas* (required)
- *gas price* (required)

On the other hand, messages are virtual objects that are never serialized [47, p. 19]. This is due to the fact that all messages can be reconstructed by replying the transaction that triggered a contract's code to run and thus, produced the message in question [2]. More, they only implicitly contain the sender's address and do not allow the gas price to be specified as is explained in the following section.

Gas

Whereas Bitcoin's scripting language is constrained to simple `true/false` evaluation of spending conditions, Ethereum's language is Turing-complete, meaning that it can

[1] Smallest unit of currency, i.e. 1 $Ether = 1x10^{18}$ Wei. Following the International System of Units, Ether's denominations have both a scientific name, as well as a colloquial name that pays homage to the great minds of computing and cryptography [13, p. 40].

[2] Messages are generated whenever a contract uses the `CALL` upcode [47, p. 19]

execute code of arbitrary and unbounded complexity [13, p. 25]. In turn, this leads to additional security and resource management problems as it cannot be predicted whether a program will terminate or run in infinite loops to effectively cause a Denial-of-Service. To combat this attacking scheme, Ethereum introduces a metering mechanism called *gas* which accounts for every instruction performed and assigns each a predetermined cost in units of gas [1] relative to the degree of computation required or burden imposed from writing to a persistent data store [13, pp. 32-33]. For simple transactions (i.e. a payment), the amount needed has been fixed at 21k units [13, p. 153].

By requiring transactions to set an upper limit (`STARTGAS`), execution will deterministically terminate as soon as the gas consumed exceeds the gas supplied [13, p. 33]. This gas allowance applies to both the transaction itself, as well as all sub-executions triggered through the encapsulated message [47, p. 19]. Logically, the total amount to be consumed by invoking a particular method can only be estimated because contracts can evaluate different conditions leading to different execution paths. In any case, EOAs are only billed for the gas actually used [2] [13, p. 154].

Additionally, the `GASPRICE` field allows an originator to set the exchange rate for each unit of gas (measured in Wei per gas unit). Analogous to other cryptocurrencies, the higher the gas price [3], the faster the transaction is likely to confirm. Still, it may also be set to zero and the transaction might even get mined during periods of low demand [13, p. 153].

With these two parameters in place, the definite transaction fee, collected by miners, can be calculated as specified in Equation 2.1 [13, p. 53] [47, p. 20].

$$fee = gas\ consumed * gas\ price \tag{2.1}$$

In this sense, gas is the fuel of Ethereum and has been purposefully separated as such to protect the system from Ether's volatility in value [13, p. 152]. At the same time, developers need to think of incentives for people to use the application since invocation of a smart-contract ultimately incurs monetary costs [40, p. 4].

[1] The current fee schedule can be found in [49, p. 25].
[2] Assert-style exceptions consume all gas available [38, p. 75]
[3] Often, the suggested gas price is calculated as the median across the last several blocks [13, p. 153]. However, gas prices are predominantly determined by miners and can therefore fluctuate unexpectedly [13, p. 54].

2.3.4 Functioning

State Transition Function

Like any (distributed) state machine, Ethereum requires a function to transition the current state (`S`) to the next final state (`S'`). This function is defined as `APPLY(S, TX) -> S'`, where `TX` represents any unprocessed (i.e. unconfirmed) transaction. Application works as following [47, p. 20]:

1. Check if transaction is well formed, contains valid signature and matches nonce
2. Calculate `FEE` = `STARTGAS` × `GASPRICE`, determine sending address from signature, subtract fee and increment nonce
3. Set `GAS` = `STARTGAS` and take off certain quantity per byte to pay for in transaction
4. Transfer value to receiving account and run specified contract code if present
5. If transaction failed due to insufficient funds or out of gas exception, revert all state changes except payment of mining fees
6. Otherwise, refund remaining gas to sender and add fee to miner's balance

Code Execution

Instead of only tracking the state of currency ownership, Ethereum tracks a general-purpose, key-value data store [13, p. 28]. More complex state transitions (i.e. not a payment) can be achieved by loading the code stored for a specific contract account into memory and running it on an emulated computer called the Ethereum Virtual Machine (EVM) [13, p. 57]. EVM code is written in a low-level, stack-based bytecode language, generated from compiling one of the many high-level languages available [1] (e.g. LLL, Solidity, Vyper) [47, p. 22] [13, p. 29].

In general, code execution is an infinite loop that repeatedly carries out the operation at the current program counter until the end of code is reached or an `ERROR`, `STOP` or `RETURN` instruction is detected. During execution in the EVM, code may access the value, sender and data of an incoming message, in addition to basic block header data and the global account state. Furthermore, operations have access to three types of space in which to store data [47, p. 22]:

- stack (follows LIFO-principle with push and pop methods)

[1] A curated list of compatible smart-contract languages can be found at [10].

- memory (infinitely expandable byte array)
- storage (persistent key-value store)

Lastly, it must be mentioned that the EVM is modelled as a completely isolated, sandboxed process preventing access to the network and filesystem [38, p. 19]. This has the negative effect of rendering most traditional APIs useless [33, p. 46].

Block Validation

Ethereum, by definition, is a system that allows concurrency of operations but enforces a singleton state at each mined block [13, p. 151]. To ensure that this state (also applies to forks) represents a valid state, the following block validation algorithm must be carried out by each node upon downloading a foreign block [13, p. 57] [47, pp. 23]:

1. Check if previous block referenced exists and is valid
2. Check that timestamp is greater than previous but less than 15 minutes in future
3. Check that block number, difficulty and various other low-level Ethereum-specific parameters are valid
4. Check that proof-of-work on block is valid
5. Set `s[0]` = state of previous block
6. Set `TX` = block transaction list (with n transactions). For each `i` in `0...n-1`, set `s[i+1] = APPLY(S[i], TX[i])`. Return an error, if any iteration returns an error (comp. State Transition Function) or total gas consumed exceeds the `GASLIMIT` [1].
7. Set `s_final` = `s[n]` with block reward paid to miner
8. Verify that merkle tree root of `s_final` is equal to root provided in header

This procedure also directly answers the question where contract code will be executed in terms of physical hardware. Namely, on all nodes that download and validate blocks containing transactions whose data field matches one of the methods of the recipient's contract code [47, p. 24]. a

[1] Similar to Bitcoin's block size, the `GASLIMIT` regulates the maximum number of transactions that can be included within a block [11] [49, p. 8].

3 Concept

3.1 Solution Overview

3.1.1 Functional Requirements

3.1.2 Non-Functional Requirements

3.2 Evaluation Framework

3.3 Architecture

3.3.1 Assumptions

4 Implementation

5 Evaluation

6 Conclusion and Discussion

7 Summary

8 Outlook

8.1 Enhancements and Additions

8.2 Adoption and Scalability

8.3 Additional Fields of Application

Bibliography

[1] 2014. URL: http://www.johnstonslaw.org/.

[2] URL: https://www.duh.de/mehrweg-klimaschutz0/einweg-plastikflaschen/.

[3] URL: https://www.reiling.eu/news.php.

[4] URL: https://github.com/dmehrotra/fuck-off-aws.

[5] URL: https://web3.foundation/.

[6] URL: https://stackexchange.com/sites#technology-traffic.

[7] URL: https://bp.eosgo.io/explore/?search_keywords=&job_category%5B%5D=block-producer&tab=search-form&type=place.

[8] URL: https://neo.org/consensus.

[9] URL: https://www.ethernodes.org/network/1.

[10] URL: https://github.com/s-tikhomirov/smart-contract-languages.

[11] URL: https://en.bitcoin.it/wiki/Block_size_limit_controversy.

[12] Jake Adelstein. *Japan's Financial Regulator Is Pushing Crypto Exchanges To Drop 'Altcoins' Favored By Criminals*. Forbes. Apr. 2018. URL: https://www.forbes.com/sites/adelsteinjake/2018/04/30/japans-financial-regulator-is-pushing-crypto-exchanges-to-drop-altcoins-favored-by-criminals/.

[13] Andreas Antonopoulos and Gavin Wood. *Mastering Ethereum: Building Smart Contracts and DApps*. Ed. by Mike Loukides. First Edition. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., Apr. 2018. ISBN: 9781491971949. URL: https://www.safaribooksonline.com/library/view/mastering-ethereum/9781491971932/.

[14] James Ball, Julian Borger, and Glenn Greenwald. *Revealed: how US and UK spy agencies defeat internet privacy and security*. The Guardian. Sept. 2013. URL: https://www.theguardian.com/world/2013/sep/05/nsa-gchq-encryption-codes-security.

[15] Russel Brandom. *Using the internet without the Amazon Cloud*. The Verge. July 2018. URL: https://www.theverge.com/2018/7/28/17622792/plugin-use-the-internet-without-the-amazon-cloud.

[16] Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit. *Mehrweganteile am Getränkeverbrauch nach Getränkebereichen in den Jahren 1991 bis 2013 (in %) in der Bundesrepublik Deutschland*. 2015. URL: https://www.bmu.de/fileadmin/Daten_BMU/Bilder_Infografiken/verpackungen_mehrweganteile_bf.pdf.

[17] *Bundesweite Erhebung von Daten zum Verbrauch von Getränken in Mehrweg- und ökologisch vorteilhaften Einweggetränkeverpackungen für die Jahre 2014 und 2015*. Texte 52/2017. Dessau-Roßlau: Umweltbundesamt, Feb. 2017.

[18] Vitalik Buterin. *DAOs, DACs, DAs and More: An Incomplete Terminology Guide*. Ethereum Foundation. May 2014.

[19] Vitalik Buterin. *Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform*. ethereum.org, 2014.

[20] Julia Chatterley, Scarlet Fu, and Joseph Lubin. *Ethereum Co-Founder on Bitcoin and Blockchain Tech*. Bloomberg. Dec. 2017.

[21] *Dosenpfand: Handel will die Wahrheit nicht wissen*. SPIEGEL ONLINE GmbH. Sept. 2001. URL: http://www.spiegel.de/politik/deutschland/dosenpfand-handel-will-die-wahrheit-nicht-wissen-a-156398.html.

[22] *Dritte Verordnung zur Änderung der Verpackungsverordnung*. Bundesgesetzblatt Teil I 29/2005. Bundesanzeiger Verlag, May 2005.

[23] Fritz Flanderka, Haucke Schlüter, and Joachim Quoden. *Verpackungsverordnung: Kommentar*. Praxis Umweltrecht 8. Heidelberg: C.F. Müller Verlag, 1999.

[24] Fritz Flanderka, Clemens Stroetmann, and Frank Sieberger. *Verpackungsverordnung: Kommentar für die Praxis unter vollständiger Berücksichtigung der 5. Änderungsverordnung*. 3. Auflage. Heidelberg: C.F. Müller Verlag, 2009.

[25] *Fünfte Verordnung zur Änderung der Verpackungsverordnung*. Bundesgesetzblatt Teil I 12/2008. Bundesanzeiger Verlag, Apr. 2008.

[26] Ian Grigg. *EOS - An Introduction*. block.one. July 2017.

[27] Uta Hartlep and Rainer Souren. *Recycling von Einweggetränkeverpackungen in Deutschland: Gesetzliche Regelungen und Funktionsweise des implementierten Pfandsystems*. Ilmenauer Schriften zur Betriebswirtschaftslehre 2/2011. Ilmenau: Verl. proWiWi, 2011. ISBN: 978-3-940882-27-1. URL: http://hdl.handle.net/10419/55711.

[28] David Johnston et al. *The General Theory of Decentralized Applications, Dapps*. https://github.com/DavidJohnstonCEO/DecentralizedApplications. Feb. 2015.

[29] Everett Muzzy and Jeff Gillis. *The State of the Ethereum Network*. ConsenSys. June 2018. URL: https://media.consensys.net/the-state-of-the-ethereum-network-949332cb6895.

[30] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Bitcoin.org, Oct. 2008.

[31] *NEO Entered the Era of Decentralization*. neo.org. July 2018. URL: https://neo.org/blog/details/4089.

[32] Albrecht Patrick et al. *Mehrweg- und Recyclingsystem für ausgewählte Getränkeverpackungen aus Nachhaltigkeitssicht*. PricewaterhouseCoopers AG WPG. June 2011.

[33] Siraj Raval. *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology*. Ed. by Tim McGovern. First Edition. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., Aug. 2016. ISBN: 9781491924549. URL: https://www.safaribooksonline.com/library/view/decentralized-applications/9781491924532/.

[34] T. Rummler and W. Schutt. *Verpackungsverordnung: Praxishandbuch mit Kommentar*. Hamburg, 1991.

[35] Niklas Sauer. *File Auditing: A Blockchain Based Approach*. Sept. 2016.

[36] Larry Seltzer. *Serverless computing explained*. Hewlett Packard Enterprise. July 2018. URL: https://www.hpe.com/us/en/insights/articles/serverless-computing-explained-1807.html?jumpid=em_khs4py1ic5_aid-510366305#.

[37] Alexander Smoltczyk and Matthias Geyer. "Die Dosenrepublik". In: *Der Spiegel*. 32/2003. Aug. 2003.

[38] *Solidity Documentation: Release 0.4.25*. Aug. 2018.

[39] Melanie Swan. *Blockchain: Blueprint for a New Economy*. Ed. by Tim McGovern. First Edition. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., Feb. 2015. ISBN: 9781491920497. URL: https://www.safaribooksonline.com/library/view/blockchain/9781491920480/.

[40] *The Growth of Blockchain Technology*. stackoverflow.

[41] *Top 100 Tokens By Market Capitalization*. CoinMarketCap. Aug. 2018. URL: https://coinmarketcap.com/tokens/.

[42] *Verordnung über die Vermeidung und Verwertung von Verpackungsabfällen (Verpackungsverordnung - VerpackV)*. Bundesgesetzblatt Teil I 56/1998. Bundesanzeiger Verlag, Aug. 1998.

[43] *Verordnung über die Vermeidung von Verpackungsabfällen (Verpackungsverordnung - VerpackV)*. Bundesgesetzblatt Teil I 36/1991. Bundesanzeiger Verlag, June 1991.

[44] Cora Wacker-Theodorakopoulos. "Pflichtpfand: Wirkungsloses Instrument". In: *Wirtschaftsdienst* 88.9 (2008), p. 558. DOI: 10.1007/s10273-008-0838-y. URL: http://hdl.handle.net/10419/42993.

[45] Cora Wacker-Theodorakopoulos. *Zehn Jahre Duales System Deutschland*. 2000. URL: http://hdl.handle.net/10419/40580.

[46] Sha Wang and Jean-Philippe Vergne. "Buzz Factor or Innovation Potential: What Explains Cryptocurrencies' Returns?" In: *PLOS ONE* 12.1 (Jan. 2017), pp. 1–17. DOI: 10.1371/journal.pone.0169556. URL: https://doi.org/10.1371/journal.pone.0169556.

[47] *White Paper: A Next-Generation Smart Contract and Decentralized Application Platform*. GitHub. URL: https://github.com/ethereum/wiki/wiki/White-Paper.

[48] Rachel Wolfson. *Vitalik Buterin On The State Of Ethereum, The Future Of Blockchain And Google Trying To Hire Him*. Forbes. Aug. 2018. URL: https://www.forbes.com/sites/rachelwolfson/2018/08/15/vitalik-buterin-on-the-state-of-ethereum-the-future-of-blockchain-and-google-trying-to-hire-him/amp/.

[49] Gavin Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger Byzantium Version*. GitHub. June 2018.

[50] Dylan Yaga et al. *Blockchain Technology Overview*. National Institute of Standards and Technology. Jan. 2018.