Wilhelm Büchner Hochschule
University of Applied Sciences
Hilpertstraße 31
64295 Darmstadt

Fachbereich Wirtschaftsingenieurwesen und

Technologiemanagement

Blockchain tokens: a review

H-ENSV01

Name: Lucas Konstantin Bärenfänger
blockrookie.com

Anschrift: Am Trauben 54
63303 Dreieich

Matrikelnummer: 903656

Studiengang: M.Sc. Technologie- und Innovationsmanagement

Betreuer: Michael Best

Abgabe: 20.07.2020

Repository: https://github.com/blockrookie/blockchain-tokens-paper

Eidesstattliche Erklärung

Name: Lucas Konstantin Bärenfänger

Matrikelnummer: 903656

Hiermit erkläre ich, dass ich diese Arbeit selbstständig abgefasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich bin mit einer Plagiatsprüfung einverstanden.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Dreieich, 20.07.2020

.....

Ort, Datum Unterschrift

Table of contents

1. Introduction

Nowadays, blockchain technology is discussed in the context of a plethora of use cases. While originally conceived within the scope of Bitcoin in 2008—which aims to provide a decentralized payment network—its applications now go far beyond cryptocurrency. In 2013, Ethereum was proposed, popularizing the concept of smart contracts to enable general-purpose use of blockchain technology. This enabled the 2017-rise of initial coin offerings (ICOs), a blockchain-based form of initial point offerings (IPOs), used predominantly by start-ups to raise funds. More recently, the application of blockchain technology with regards to the Internet of Things (IoT) is explored, aiming to represent billions of IoT devices on blockchain-based systems. Furthermore, central bank digital currency (CBDC) is a hotly debated issue, with central banks investigating if and how legal tender can be issued digitally.

One could argue that all these applications of blockchain technology hinge on the notion of blockchain tokens. Therefore, it is the goal of this paper to provide a thorough introduction into the topic of tokens. It aims to equip the reader with the knowledge necessary to follow, evaluate and participate in the discourses mentioned above. To this end, this paper explicitly avoids crude analogies and simplifications—which, unfortunately, are all too common in the blockchain domain. Instead, it presents and examines primary sources, e.g., white papers of the technologies that contributed to the state of the art.

Section 2 describes Bitcoin, the very first blockchain-based system. It is crucial to understand its inner workings, as any development in the blockchain domain is commonly laid out in reference to Bitcoin. Section 3 introduces the Ethereum platform and its notion of smart contracts. While smart contracts existed before Ethereum, it was Ethereum that popularized them by providing sophisticated tools to employ them. Section 4 presents relevant aspects of what is referred to as the Liechtenstein Blockchain Act. Beyond a technological definition of tokens, the Liechtenstein Blockchain Act, being the first European legislation to introduce legal certainty in the blockchain domain, gives an account of the concept from a legal perspective. Section 5 describes the Stellar payment network. Stellar is a blockchain-based system geared towards—but not limited to—payments. It does away with many of the shortcomings of early such systems like Bitcoin and Ethereum. It can be argued that Stellar represents the state of the art of blockchain technology and will be of crucial importance for further developments in the area of tokens. Section 6 summarizes the results.

2. Blockchain as first introduced by Bitcoin

This section introduces the concept of blockchain using the example of Bitcoin, the very first blockchain-based system. In the course of this, terms and concepts relevant in the blockchain domain are defined and explained. This introduction is based on the original Bitcoin paper [1], which was published under the pseudonym Satoshi Nakamoto in 2008. It is not intended, however, to cover all aspects of Bitcoin but merely to distill what is relevant in the context of blockchain in general. Thus, parts specific to cryptocurrencies are omitted when possible.

The original Bitcoin paper proposes a "peer-to-peer version of electronic cash," which allows for the transfer of a **coin** within a decentralized network, i.e., without relying on a trusted party.

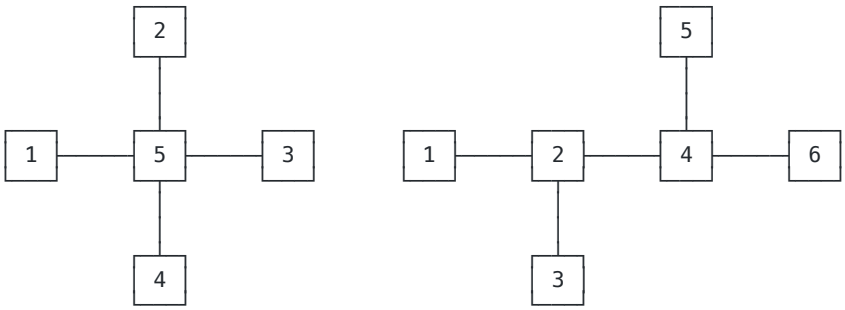

Figure 1: A centralized vs. a decentralized network topology

Public-key cryptography, specifically, digital signatures[1], "provide part of the solution" proposed in the Bitcoin paper. A **transaction** represents the transfer of a coin from one owner to the next. A payer uses their private key to sign a transaction, and a payee may use the public key of a payer to verify a transaction's signature. Thus, public-key cryptography ensures that a coin can only be spent by its righteous owner.

[1] In a system based on public-key cryptography, a participant generates two keys, a private key and a public key. The private key must be kept secret, while the public key may be shared freely. In the context of this introduction, such keypairs are used to create and verify digital signatures. A person—commonly called Alice—uses their private key to create a digital signature for a message. Another person—commonly called Bob—uses Alice's public key to verify that a signature was, in fact, created with Alice's private key, which, presumably, is only accessible to her. Since a signature is only valid for the exact message it was created for, it also guarantees that a message has not been tampered with.

However, this does not prevent an owner of a coin from double-spending it. In order to double-spend a coin, an owner creates and signs two or more transactions, each of which transfers ownership of the coin. In a centralized network, these transactions would be submitted to one trusted party, which would acknowledge one of them and discard the other(s). The proposed solution is, however, based on a decentralized network. Thus, the transactions are—like all transactions—not submitted to a trusted party but broadcast to all network nodes. To avoid double-spending, all nodes must acknowledge the same transaction and discard the other(s). It is "the earliest transaction [...] that counts." However, within a peer-to-peer network, nodes are not guaranteed to receive transactions in the same order. Therefore, nodes have to agree upon "a single history of the order in which [transactions] were received." A solution that facilitates such agreement, i.e., a solution to the **double-spending problem**, is the main contribution of the Bitcoin paper.

The proposed solution begins with the definition of a generic "timestamp server." Items to be timestamped, e.g., transactions, are grouped to form a "block of items." A block is hashed[2] along with the timestamp of its preceding block. The resulting hash constitutes the block's timestamp, which is published by the timestamp server. Since both a block and the timestamp of its predecessor must exist in order to be hashed, a timestamp proves just that—their existence at the time of timestamping. Obviously, timestamps form a conceptual chain, as each timestamp is based on the hash of its preceding timestamp. Thus, each timestamp is "reinforcing" the ones preceding it, as manipulating a block not only requires recalculating its timestamp but all of the subsequent timestamps.

[2] A hash function maps a typically large, variable-length value to a typically small, fixed-length value. The output of a hash function is called a hash. A cryptographic hash function is expected to be a one-way function that is, among other things, deterministic and collision resistant. A one-way function is quick to compute but practically impossible to invert, i.e., it is practically impossible to infer from the hash the input value. Deterministic means that hashing the same input value should always result in the same hash. Collision resistant means that it is practically impossible to find two distinct input values that have the same hash.

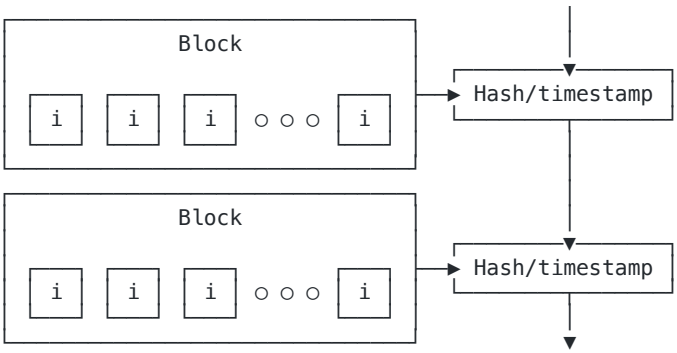

Figure 2: Timestamp server
(based on figure 2 of the Bitcoin paper)

Assume each node of the decentralized network would run the timestamp server. A node would group transactions it received into blocks, timestamp them and publish the timestamps. The resulting chain of block timestamps would represent the history of the order of transactions as perceived by the individual node. In order to solve the double-spending problem, however, nodes still have to agree upon one such chain in order to establish an overall history of the order of transactions.

To this end, the Bitcoin paper proposes a “distributed timestamp server on a peer-to-peer basis.” It is based on a challenge to find a special number, which is referred to as “nonce.” The nonce has to be such that when hashed along with a block and the hash of its preceding block, the resulting hash starts with a set number of zero bits. Since cryptographic hashing functions are one-way functions, there is no way to compute the nonce—it has to be guessed. Thus, an arbitrary number is incremented until a hash that leads with a number of zero bits is obtained. Once found, the nonce becomes part of the respective block, which is broadcast to the network. The nonce is also referred to as **proof of work**, as it can be thought of as proof that work, i.e., “CPU effort,” has been expended in order to find it. The necessary amount of work depends on the number of zero bits that a hash is expected to start with. Therefore, the **difficulty** within this proof-of-work-based system can be adjusted by changing this requirement. As with the previous timestamp server, blocks form a chain, as each block’s proof of work is based on the hash of its preceding block. Also as before, each block hardens the ones preceding it, as manipulating a block not only requires to expend the work to guess a proof of work for it but to do so for all subsequent blocks.

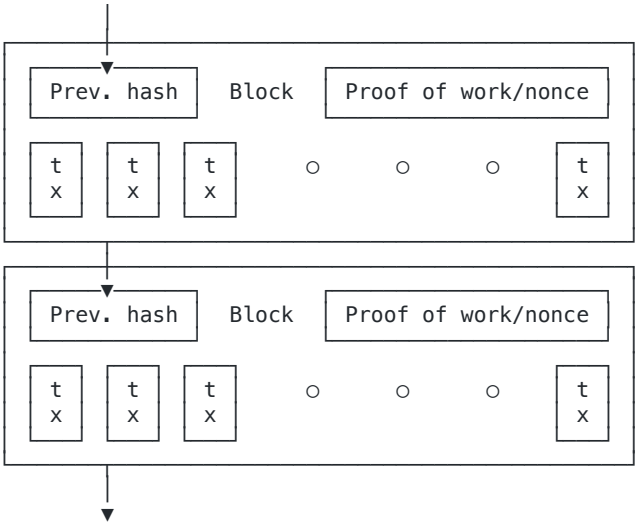

Figure 3: Distributed timestamp server with proof of work
(based on figure 3 in the Bitcoin paper)

Each node of the decentralized network runs the distributed timestamp server. As mentioned, if a node happens to find a proof of work for a block, it announces the block—which includes that nonce—to the network. A node accepts a block it receives from the network if the block's proof of work and all of its transactions are valid, e.g., do not constitute double-spending. A node "express[es]" its "acceptance" of a block by adding it to its chain and then proceeding to create the next block on top of it.

A node always acknowledges the longest chain it learns from to be the correct one. In case a node learns about a chain that is longer than the one it is currently building upon, it switches towards creating a block on top of that longer chain. Since a node's chain of blocks represents the history of the order of transactions as perceived by that node, and since all nodes agree on one such chain, i.e., the longest chain, they, in fact, agree on one history of the order of transactions, which constitutes a solution to the double-spending problem.

Not only does the proposed system solve the double-spending problem, it also constitutes a very robust solution. Assume an attacker would attempt to tamper with one of the transactions of a block within the chain. Obviously, the attacker would have to redo the proof of work for the block containing the transaction as well as for all subsequent blocks. Moreover, in order for the entire network to accept the manipulated chain, it would have to become the longest chain. To this end, it would have to grow faster than the "honest chain"—to which all the "honest nodes" contribute—and at some point overtake it. However, as the longest chain "[comes] from the largest pool

of CPU power," an attacker would have to control "more CPU power than all the honest nodes" combined in order to accomplish this.

The question remains why one would choose to join the decentralized network that forms the proposed peer-to-peer version of electronic cash. After all, work, i.e., CPU power, has to be expended, which consumes electricity and, thus, incurs costs. In the Bitcoin paper, two incentives are described. Firstly, a node may add a "special transaction" to the beginning of a block. This transaction transfers ownership of a new coin to the node—which is how coins come into existence in the first place. Secondly, a transaction may contain a "transaction fee." It serves as incentive for a node to include the transaction in a block. Thus, a node that succeeds in finding a proof of work for a block is rewarded with a new coin as well as the fees of the block's transactions. Note that in order to prevent inflation, at some point no more new coins are brought into circulation, leaving transaction fees to be the sole incentive for participation.

At this point, important terms from the blockchain domain have been introduced: coin, transaction, double-spending problem, proof of work and difficulty. There are, however, terms that are yet to define, since they are not mentioned in the original Bitcoin paper. Most importantly, the term **blockchain** itself is not used within the paper. In a narrow sense, it obviously refers to the previously described "chain of blocks," i.e., to the data structure used to represent the transaction history. In a broader sense, the term blockchain refers to a class of systems which enable the transfer of value within decentralized networks. The term **ledger** is commonly used to refer to the blockchain, as a data structure that represents a transaction history exactly corresponds to what accountants call a ledger. In the context of blockchain-based systems, a **consensus algorithm** is an algorithm that enables the nodes of a decentralized network to agree on—or, reach consensus on—one ledger. It oftentimes is the distinguishing feature of such a system. The consensus algorithm proposed in the Bitcoin paper is, as mentioned, referred to as proof of work. Lastly, a node that engages in the process of creating new coins by bundling transactions into blocks and attempting to find a proof of work for them is commonly called a **miner**, in analogy to mining gold.

3. Smart contracts as popularized by Ethereum

This section introduces the concept of smart contracts using the example of smart contracts in Ethereum, as they are commonly discussed in reference to Ethereum's notion of them. This is, arguably, because Ethereum was the blockchain-based system that popularized smart contracts. The concept is introduced theoretically as well as practically with code examples.

As described in the previous section, blockchain-based systems enable the nodes of decentralized networks to reach consensus. In the case of the Bitcoin network, nodes agree upon a single transaction history in order to facilitate a peer-to-peer version of electronic cash. Blockchain-based systems can, however, be leveraged for a variety of use cases—beyond currency. In a 2015-talk [2], Vitalik Buterin, co-founder of Ethereum, names examples of such alternative applications, including asset issuance, domain registration and voting. However, Buterin criticizes that early blockchain-based systems only ever implement one such use case, respectively, and therefore considers them "single-purpose tools [...] designed around one particular application." Obviously, he goes on to introduce Ethereum as a blockchain-based system that is general-purpose and, thus, enables "people [to] build whatever they want on top of it." It is to be noted that while Ethereum aims to enable general-purpose use of blockchain capabilities, these capabilities are still based on proof of work, just like in Bitcoin.

In order to highlight how Ethereum differs from Bitcoin and how it enables the aforementioned generality, the Ethereum white paper [3] describes both the Bitcoin system and the Ethereum system as formal abstractions. More precisely, it defines "the Bitcoin ledger [...] as a state transition system." State is represented by the "ownership status" of all coins and a state transition is triggered by a transaction. Accordingly, a "state transition function" `APPLY(S, TX) = S'` is "formally define[d]." Applying the function to a (current) state `S` and a transaction `TX` results in a new state `S'`. Ethereum aims to extend this model towards "what is essentially the ultimate abstract foundational layer." To this end, users may define **smart contracts** "to encode arbitrary state transition functions" and "transaction formats." State in Ethereum is "made up of [...] accounts," of which there are two types: "externally-owned accounts," i.e., user accounts, which are "controlled" by private keys, and "contract accounts," which are controlled by the source code of the account's contract. Accounts are made up of fields, including an account's balance of ether, i.e., Ethereum's native cryptocurrency,

as well as—in the case of contract accounts—the source code of an account's smart contract and storage where the smart contract's data lives.

With regards to Ethereum's actual implementation, smart contracts are described as executable programs "which automatically move digital assets according to arbitrary pre-specified rules." The execution of a smart contract is triggered by a transaction sent to its address. As to where—"in terms of physical hardware"—a smart contract is executed, it is noted that the execution process is "part of the block validation algorithm." Thus, a smart contract is executed by each network node that "downloads and validates" the block that contains a transaction that triggers its execution. Consequently, when triggered, a smart contract is eventually executed by each individual node. A block in Ethereum contains—like a Bitcoin block—"the transaction list." Beyond that, it contains "the most recent state" as well as "the block number and the difficulty."

It is noted that the Bitcoin system actually implements a "weak version" of smart contracts, as a coin cannot only be owned by a public key but also by a "more complicated script," which has to be "satisfie[d]" by any transaction that attempts to spend the coin. However, Bitcoin's scripting language is considered to have "limitations," which are, obviously, remedied in Ethereum. These limitations make for a good set of criteria to evaluate a blockchain-based system's capabilities with regards to smart contracts:

- The Bitcoin scripting language lacks **Turing completeness**[3], with "the main category that is missing" being loops. This is a deliberate design choice to make it impossible to define non-halting smart contracts[4], e.g., smart contracts that contain infinite loops, since allowing such contracts would essentially

[3] A Turing machine—first described by Alan Turing in 1936—is a formal model of computation. As such, it provides a formal definition of what is computable and what is not. A problem is computable if a Turing machine can be constructed to solve it. A system is Turing-complete if it is able simulate any Turing machine. In other words, a system, e.g., a programming language, is Turing-complete if it can be used to solve any computable problem.

[4] The halting problem describes the problem of determining for any given algorithm whether or not it will terminate. In other words, it describes the problem of telling whether an arbitrary computer program will halt or keep on running infinitely. In 1936, Alan Turing proved that no Turing machine can exist that decides this question for any given algorithm. Thus, given an arbitrary computer program, it is impossible to compute whether it will halt or keep on running infinitely.

open up the entire system to denial-of-service attacks (DoS attacks). Ethereum, on the other hand, does feature a Turing-complete programming language, in which such smart contracts could be expressed. However, a transaction that triggers a smart contract's execution specifies "the maximum number of computational steps the transaction execution [i.e., the contract execution] is allowed to take" and how much ether the transaction's sender pays "per computational step." Thus, a contract that contains infinite loops would naturally run out of paid-for computational steps and would therefore not execute infinitely.

- Bitcoin's scripting language is limited due to its "[v]alue blindness." As mentioned, a coin can be owned by a script, however, this script cannot provide "fine-grained control over the amount that can be withdrawn." Ethereum, conversely, features "**value awareness**."
- Bitcoin scripts are "blockchain-blind," e.g., unaware of "the timestamp and previous block hash." Obviously, Ethereum smart contracts feature "**blockchain awareness**."
- Bitcoin scripts cannot "keep [...] internal **state**," whereas smart contracts in Ethereum have a "key/value store to keep track of persistent variables" at disposal.

There are several high-level programming languages which can be used to develop Ethereum smart contracts. At the time of this writing, Solidity, a JavaScript-like programming language, is the default choice. The code examples in the Ethereum white paper—which was originally published in 2013—are, however, written in Serpent, a Python-like programming language, which has since been deprecated [4]. All of Ethereum's high-level programming languages compile down to "Ethereum virtual machine code," or, for short, "EVM code," which is what is executed during block validation.

The following code example was taken from the Ethereum white paper. It demonstrates how to implement "a token system in Serpent." The concept of tokens will be defined in the next section—for now, a token can be regarded as an alternative coin next to a platform's native cryptocurrency that is also maintained on said platform's blockchain.

```
def send(to, value):
    if self.storage[msg.sender] >= value:
        self.storage[msg.sender] = self.storage[msg.sender] - value
        self.storage[to] = self.storage[to] + value
```

Listing 1: A smart contract implementing a token system (taken from the Ethereum white paper)

This smart contract "fundamentally" represents a "database with one operation," i.e., the transfer of `value` from A (denoted by `msg.sender`) to B (denoted by `to`). If A's funds are sufficient, `value` is subtracted from A's balance (denoted by `self.storage[msg.sender]`) and added to B's (denoted by `self.storage[to]`). Obviously, the contract account's storage is leveraged to persist everyone's balances.

As mentioned, the first blockchain-based system, Bitcoin, aims to decentralize electronic cash to eliminate the need for a trusted party. As Ethereum sets out to generalize this principle, it enables users to define smart contracts to decentralize applications that would traditionally require a trusted party. With this in mind, the following smart contract is designed to replace a trusted party that evenly divides up incoming currency, i.e., ether, between three beneficiaries. This code example—which has been created specifically for this paper—is written in Solidity, i.e., the high-level programming language that is recommended for Ethereum smart contract development at the time of this writing.

```
pragma solidity ^0.6.0;

contract SplitPot {
  address payable[] beneficiaries = [
    0x498898b3F52DAba1bB304a4b4D2EA31a111484B1,
    0xAcb19c763EB67ea757Efd8Cd8b6ecceb28F1284B,
    0xD5d3f3650C4DdE7B8034671129443A596Ce8ed57
  ];

  receive() external payable {
    uint individualAmount = msg.value / beneficiaries.length;

    for (uint i = 0; i < beneficiaries.length; i++) {
      beneficiaries[i].transfer(individualAmount);
    }
  }
}
```

Listing 2: A smart contract evenly dividing up ether

In the smart contract—which is called `SplitPot`—the `addresses` of the three `beneficiaries` are encoded. Whenever ether is `received` by the smart contract, the `individualAmount` that each beneficiary is entitled to is calculated by dividing the amount of received ether (denoted by `msg.value`) by the number of beneficiaries (denoted by `beneficiaries.length`), i.e., three. Then, each beneficiary is `transferred` their share. Note that this smart contract has been designed for demonstration purposes and does not represent production-grade software—its shortcomings [5] are intentionally tolerated to keep it simple and short.

4. Tokens as defined by the Liechtenstein Blockchain Act

This section introduces the concept of tokens based on the "Report [...] of the [Liechtenstein] Government [...] Concerning the Creation of a Law on Tokens and TT Service Providers [...]" [6]. As its title suggests, this report describes the "Tokens and TT Service Providers Act" (TVTG), which was enacted into Liechtenstein law January 1st, 2020. The TVTG—which is commonly referred to as Liechtenstein Blockchain Act—is Europe's first legislation aiming to regulate the blockchain domain, and, specifically, tokens. As such, the TVTG and the government report on it provide definitions and explanations of the terms and concepts relevant with regards to tokens.

The TVTG report states that at the "core of a blockchain system," there is "information," which is "attributable to one person" as well as "securely" transferrable "to another person." This information may "assume different functions," e.g., it may represent a form of electronic cash, as is the case in the Bitcoin system. In some blockchain-based systems, this information is referred to as coin, e.g., in Bitcoin, while in others, it is "named a token." Based on these remarks, it can be concluded that a **token, in a technical sense**, is an entity that is maintained and transferred within a blockchain-based system.

The TVTG report explains that the law uses the term token—rather than coin or some other term—because it expresses "independence" and "portability." The term coin arguably indicates the type of blockchain-based system, i.e., Bitcoin-like systems, and also the use case, i.e., payments, while the term token is abstract in the sense that it can be used in the context of any blockchain technology and use case. In fact, the entire TVTG was aimed to be drafted "abstractly enough to ensure that it remains applicable for subsequent technology generations." Therefore, the TVTG does not only avoid the term coin but also the term blockchain. Instead, the law refers to "transaction systems based on trustworthy technologies (TT systems)." According to the TVTG report, this captures what is essential to blockchain-based systems—"[t]he fact that trust is created by technology and not solely by organi[z]ations"—without being technologically restrictive. It is also stated that this terminology "implicitly includes" other central aspects of blockchain-based systems, namely "the use of cryptography" and "decentralization" to facilitate "trust in the integrity of the main ledger."

According to the TVTG report, “it has [...] become clear” that what “can be represented” on TT systems, i.e., what can be represented by tokens, goes well beyond electronic cash. Cited examples of further applications include “software usage rights” and “financial applications.” A “legal definition” of tokens must therefore be “a more abstract formulation” that encompasses “the full range of potential applications.” It is argued that these applications can be viewed as rights, e.g., “the right to purchase Swiss francs,” “ownership or usage rights to property,” etc. Thus, the TVTG introduces a **token, in a legal sense**, as a “legal object” that enables the representation of “all types of rights on a TT system.” A token can be regarded as a “container” that holds a right. In this sense, a bitcoin is an example of an “empty” token, i.e., a container that does not hold a claim to anything. Beyond this general account of what legally constitutes a token, the TVTG specifically defines a “**payment token**,” stating that it is “accepted to fullfill [sic] contractual obligations and therefore replace[s] legal tender in this respect.”

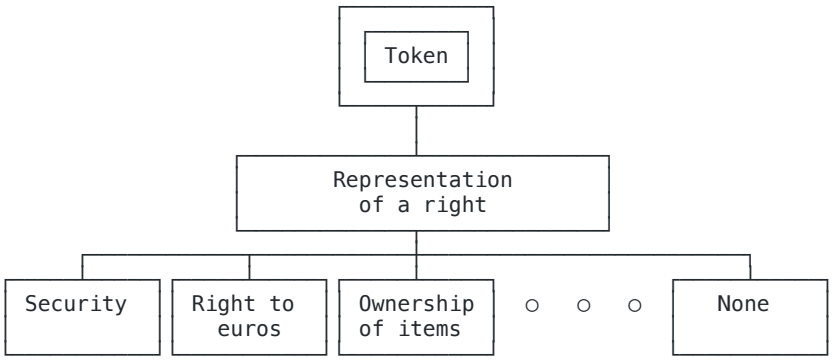

Figure 4: Token container model (based on figure 14 in the TVTG report)

This legal definition of tokens raises “various questions,” especially with regards to “the legal connection between the token and the represented right.” In order to establish a technology-neutral terminology to answer these questions, the TVTG defines further elements that are central to TT systems in an abstract fashion, including the following:

- A “TT identifier” is assigned to a person or—in the context of IoT—to a machine. It corresponds a public key.
- A “TT key” enables the “de-facto,” i.e., technical, “[d]isposal over [...] the tokens allocated to the TT identifier.” It corresponds to a private key.

- A “holder of [a] TT key” is a person or machine that “can actually [i.e., technically] dispose over the TT key.”
- A “person possessing the right of disposal over the TT key” is legally entitled to dispose over the TT key.
- A “delegate of the person holding the right of disposal” is an “independent role” that may be entrusted with the “safekeeping of TT keys.”

Note that the holder of a TT key and the person possessing the legal right of disposal are “irrefutably assume[d]” to be the same person. If, however, an unauthorized “third party” gains control of the TT key, it assumes the role of the holder of the TT key, but, obviously, does not become the person possessing the right of disposal. Hence, two distinct roles are defined. If a person possessing the right of disposal chooses to delegate that right, the delegate then also assumes the role of the person possessing the right of disposal.

It can be argued that the TVTG elegantly avoids regulating each individual use case for TT systems by, instead, identifying and regulating “TT service providers” that are crucial to a token’s life cycle. Examples of those regulated roles are the following:

- A “token generator” generates tokens by “correctly” recording “asset objects and rights in digital form.”
- A “token issuer” publicly offers tokens and is required to offer “basic information” about them.
- A “TT key Depository”/“TT token depository” is entrusted with “safekeeping” of TT keys/tokens “for clients.”
- A “physical validator” ensures synchrony “between the object and the token that represents rights to it.”
- A “TT exchange service provider” facilitates the exchange of legal tender and payment tokens.

The role of a physical validator deserves special attention because it is crucial to how the TVTG deals with the aforementioned questions regarding the legal connection between the token and the represented right. Obviously, tokens are not restricted to holding rights to “purely digital assets” but may carry “digitalized rights to physical objects.” The TVTG acknowledges that this introduces “legal uncertainty,” as the latter

can be moved in the physical world without the knowledge of the respective token owner in the digital world. With regards to this "duality" between "offline" and "online," it is the role of a physical validator to ensure correct "[i]dentification of the object of value" and of its "lawful owner," secure and appropriate "[s]torage" as well as prevention of conflicts, e.g., making sure that an object "is not already encumbered [...] by liens" in the "analogue" sphere.

Throughout the document, the TVTG report addresses many applications of TT systems. Mentioned examples of "assets traded on blockchain systems" include "rights to precious metals, [...] property and real estate and rights to use items such as cars, watches or yachts." The TVTG reports uses the term **token economy** to refer to this breadth of possible applications. The "potential of the token economy" is then described as the "ability to reproduce" the non-digital world on TT systems "in a legally certain manner," including the ability to "transmit rights [i.e., tokens] efficiently." It can be argued, however, that for economic action to happen predominantly on-chain, i.e., on TT systems, there would have to exist a representation of government-issued currency on those systems, i.e., a TT-based variant of CBDC.

5. The state of the art as represented by Stellar

This section introduces Stellar, a blockchain-based system geared specifically towards payments. Stellar can be regarded as state-of-the-art blockchain technology, as it remedies major flaws of proof-of-work-based systems. This introduction is based on the 2019-paper “Fast and secure global payments with Stellar” [7] by members of the Stellar Development Foundation (SDF), which, in the Stellar community, is commonly referred to as “the SOSP paper.” It is to be noted, though, that the white paper that originally proposed the Stellar Consensus Protocol (SCP) [8] was already published in 2016. The SOSP paper, does, however, seem to present Stellar and, particularly, SCP in a way that is easier to parse.

Stellar’s consensus algorithm is fundamentally different from Bitcoin’s and Ethereum’s, i.e., from proof of work. SCP is based on “federated byzantine agreement” (FBA), which, in turn, is a generalization of “byzantine agreement” (BA).

The SOSP paper describes “traditional” BA to assume a fixed set of `N` nodes, with `N = 3f + 1` and `f > 0`. Nodes “receive input values” and exchange messages in order to “decide on an output value among the inputs.” In other words, they exchange messages to reach consensus with regards to the input values. BA “guarantee[s] safety and some form of liveness” as long as no more than `f` nodes are faulty. The **safety** condition is satisfied if every two non-faulty nodes decide on the same value and this value is, in fact, one of the input values, whereas **liveness** applies if every non-faulty node “eventually outputs a value.” Obviously, the safety condition captures what is essential to a consensus algorithm, as requiring nodes to decide on one value is just that—requiring consensus. To reach said agreement, “nodes vote on […] values.” `2f + 1` votes form a “quorum.” Since every two quorums overlap in `f + 1` votes, they share at least one non-faulty node. Hence, there cannot be contradictory decisions.

The problem with BA is, however, that it assumes a set of previously known nodes. It can be argued that, as such, these nodes form a centralized network, as there is the need for a third party to act as gatekeeper. Thus, BA solves a fundamentally different problem than proof of work, as it only enables the nodes of a centralized network to reach consensus—not decentralized ones.

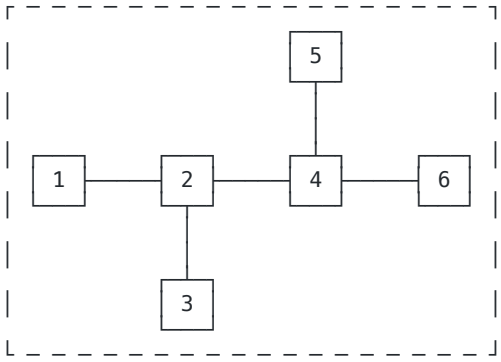

Figure 5: A variant of a centralized network topology, as a trusted party acts as gatekeeper

FBA—which was first introduced with Stellar—does, by contrast, allow for “open membership” on a peer-to-peer basis and, thus, enables nodes to “join and leave unilaterally.” In a decentralized network, however, nodes cannot universally agree on a quorum, as “two nodes may not even know of each other’s existence.” Instead, a node specifies one or more “quorum slices,” which “express with whom the node requires agreement.” This may be based on external criteria, e.g., a node may trust its local bank and, thus, choose a quorum slice consisting of nodes run by that bank.

At this point, it is observed that if every two nodes require agreement, by the property of transitivity, “we get global agreement.” If this is not the case, however, “we can get divergence.” Based on the “topology of today’s financial system,” it is hypothesized, though, that just like “the Internet” does converge—not diverge—, so will “the Stellar network.”

A quorum in FBA is defined as a “non-empty set `N` of nodes encompassing at least one quorum slice of each non-faulty member.” E.g., a node `v1` may specify the single quorum slice `{ v1, v2 }`, while nodes `v2` and `v3` may both specify the single quorum slice `{ v2, v3 }`, respectively. In this case, `{ v2, v3 }` constitutes a quorum, whereas `{ v1, v2 }` does not, since it does not encompass a quorum slice for `v2`. This example is inspired by a slightly more complex one that can be found in the Stellar internet draft [9].

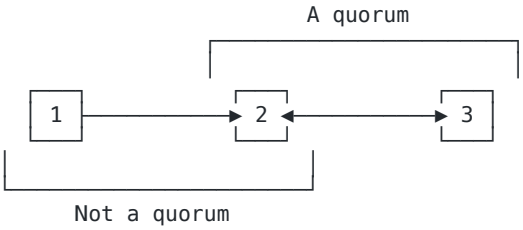

Figure 6: Quorum slices example

It is further defined that two non-faulty nodes `v1` and `v2` are considered “intertwined” if “every quorum of `v1` intersects every quorum of `v2` in at least one non-faulty node.” With regards to the safety condition, it is noted that FBA is limited to ensuring consensus “between intertwined nodes,” which is what SCP guarantees. In accordance with the aforementioned transitivity hypothesis, it is mentioned that “Stellar’s design” assumes “that the nodes people care about will be intertwined.” This assumption is then called the “Internet hypothesis.” With regards to liveness, it is stated that “SCP guarantees […] non-blocking liveness”—which is, however, beyond the scope of this paper.

The SOSP paper devotes an entire paragraph to emphasizing that in Stellar, it is “not desirable” for “safety and liveness [to] have the same fault tolerance.” Safety failure is considered tantamount to “double-spent digital money”—which would effectively render Stellar, a “payment network,” useless. Liveness failure, on the other hand, would merely increase the latency of payments—which, so the claim, in some cases “took days before Stellar” anyway.

With regards to voting, “federated voting” is introduced. Federated voting is a “three-phase protocol” in which nodes “try to agree on […] statements” about “values.” In the Stellar network, these values are sets of transactions. The three stages of federated voting are “vote,” “accept” and “confirm.” SCP can be viewed as a two-phase protocol, encompassing a “nomination” phase as well as a “balloting” phase, both of which employ federated voting. In nomination, nodes “vote to nominate multiple values” as “valid decision candidate[s],” while in balloting, nodes “prepare” and then try to “commit” those. As it is beyond the scope of this paper to dive deeper into these intricacies of SCP, it might be helpful to, instead, provide the following illustration, taken from an article on the SDF developer blog [10]. It conceptualizes the aforementioned SCP stages as a funnel, narrowing down on values until all nodes apply a single value.

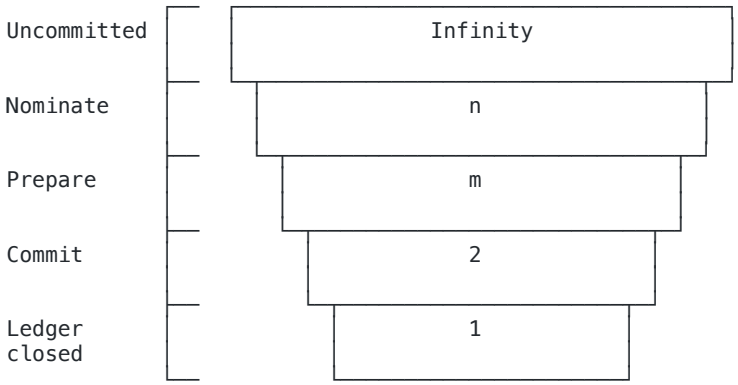

Figure 7: SCP, conceptualized as a funnel
(taken from the SDF article "Intuitive SCP")

What has been presented up to this point allows for some observations. In the following, differences between Stellar's SCP and Bitcoin's proof of work are described.

As the Bitcoin paper aims to propose a system that does away with "trusted third parties," i.e., financial institutions, it presents a trustless system—trustless in that no two nodes are required to trust each other for it to function. One could argue, though, that a node still has to trust that the majority of the Bitcoin network's processing power is in the hands of "honest nodes." In this sense, the need for trust would not have been eliminated but distributed to the network as a whole. Be that as it may, it is obvious that Stellar is based on entirely different assumptions. In fact, the Stellar network's topology is literally the result of nodes explicitly expressing trust towards the individual nodes in their quorum slices. Thus, it can be stated that SCP solves a different problem than proof of work—a weaker problem, one could argue.

As described before, Bitcoin, or, rather, proof of work, incentivizes the operation of nodes in two ways, as nodes get to keep the newly mined coins as well as the transactions fees of the blocks for which they find a proof of work. In SCP, however, there is no notion of mining, since most units of Stellar's native asset, called "lumens" (XML), came into existence at the network's genesis. Accordingly, the SOSP paper calls nodes not miners but "validators." Furthermore, it is clearly stated that "[v]alidators are not compensated by fees." Thus, Stellar lacks both of Bitcoin's incentives for running nodes—doing so is not rewarded with cryptocurrency, i.e., lumens, whatsoever. One could argue that this leads to more centralization, as less people might be inclined to operate a node and validate transactions.

While these observations highlight what can be considered drawbacks, it must be pointed out that these result from design choices that enable great advantages as well. The following advantages are of critical importance with regards to payment tokens and redeemable tokens in general.

When a transaction is submitted to the Bitcoin network, it has to be included into a block for it to become part of the ledger. As hinted at in the Bitcoin paper, the difficulty of today's operational Bitcoin network is continuously adjusted so that a block is mined approximately every 10 minutes. Once a transaction is part of a block, it is said to have reached one confirmation. With each block that is built on top of that block, the number of confirmations increases. Since a node may work on one blockchain for some time and then discard it as it learns from another, longer blockchain—which is how network convergence is ensured in Bitcoin—, many vendors typically require at least six confirmations[5] before accepting a transaction as payment and subsequently releasing purchased goods. This is based on the reasoning that the longer the blockchain is that contains a transaction, the less likely it is that a longer one comes along that does not contain it. Consequently, one could argue that, for all practical purposes, the Bitcoin network faces a **latency** of approximately 60 minutes. The Stellar white paper, on the other hand, lists "[l]ow latency" as a "key" property, stating that, in Stellar, consensus can be reached "at timescales humans expect for web or payment transactions." The SOSP paper further specifies that "the system runs SCP at 5-second intervals," adding that this is even "far slower than typical applications of Byzantine agreement."

With regards to Bitcoin, it must be emphasized that while with each additional confirmation the probability of a transaction being reversed drops exponentially, this probability never goes down to zero. It is hard to overstate that there is always the probability of the public history of transactions being reversed. Given enough time, an attacker who controls the majority of the network's CPU power will be able to reverse any transaction, no matter how many times it was confirmed. To this end, the attacker

[5] The practice of awaiting exactly six confirmations is, in fact, fairly random. From the calculations section of the Bitcoin paper, it can be inferred that if an attacker who controls 10% of the network's CPU power aims at reversing a transaction that is part of a block on top of which five blocks have been built, i.e., a block with six confirmations, their probability of success would be less than 0.1%. It must be pointed out, however, that waiting for confirmations makes sense even if the absence of attackers is assumed, as the transaction history might be rewritten as part of network reconvergence.

simply starts working on an alternative blockchain, which will at some point be longer than the honest chain and, thus, be adopted by the network. Therefore, the Bitcoin system lacks **finality**, i.e., its transaction history is never final in the sense that it cannot be reversed. The SOSP paper, on the other hand, lists “[i]ssuer-enforced finality” as one of three “goals” achieved by the Stellar system. As an SDF’s article on the issue [11] explains, “there is no branching [i.e., divergence] in SCP”—assuming, of course, the previously described Internet hypothesis holds true. Accordingly, nodes never have to “switch over” to another blockchain as a measure to achieve network reconvergence. In fact, as nodes “add new blocks to their ledger history, they will never go back and change them.”

While, with regards to payments, it is obvious that latency in the tens of minutes is impractical, to say the least, the absence of finality in proof-of-work-based systems has even more grave implications, especially in the context of redeemable tokens and their issuance. In fact, finality, in Stellar, is considered issuer-enforced, as the property of finality is of utmost importance in the context of token issuance. Assume the European central bank (ECB)—or anyone, for that matter—were to issue euro tokens, i.e., tokens that each hold a claim to one euro. To redeem such a euro token for a real euro, one would have to submit a transaction transferring the euro token back to the ECB. In the case of Stellar, the ECB could safely hand out a real euro as soon as said transaction is recorded on the ledger that is maintained by the ECB’s—i.e., the token issuer’s—node. In a proof-of-work-based system, this would not be safe, as the blockchain maintained by the token issuer’s node might be switched out against a different, longer blockchain that might not contain the transaction in question. Thus, the ECB would always be at risk to hand out real euros without owning the associated euro tokens.

In conclusion, one could argue that Stellar, tackling a problem that is only slightly weaker than what proof-of-work-based systems solve while featuring crucial—previously unsupported—properties, i.e., low latency and finality, represents a big step towards an on-chain economy, i.e., a token economy. This is only boosted by innovative features that are beyond the scope of this paper, e.g., “[c]ross-issuer atomicity,” allowing users to “atomically exchange and trade tokens from multiple issuers,” support for KYC and AML operations, etc.

6. Conclusion

In the previous sections, primary sources that are relevant to the topic of blockchain tokens were examined. In section 2, the very first blockchain-based system, Bitcoin—as described in the original Bitcoin paper—, was presented. It was revealed that the Bitcoin system or, rather, its consensus algorithm proof of work, represents the first known solution to the so-called double-spending problem, enabling the nodes of a decentralized network to reach consensus on one history. In a broader sense, blockchain-based systems have been described as systems that enable the transfer of value in such networks. In section 3, smart contracts were formally defined as vehicles to enable user-defined state transition functions and transaction formats within blockchain-based systems. On a more practical account, smart contracts were considered executable programs that automatically transfer value based on certain conditions. Two code examples were provided, including one in which a simplistic token system was implemented. In section 4, the Liechtenstein Blockchain Act was reviewed to provide a non-technical, legal account of tokens. A token was specified as a digital container that holds a legal right. Among the terms and roles defined by the legislation, the role of a physical validator was covered more extensively, as it is a physical validator that is sought to ensure synchrony between the physical and the on-chain world. In section 5, the Stellar network and its consensus algorithm, SCP, were reviewed. It became clear that SCP solves a different, arguably weaker problem than proof of work. By featuring low latency and issuer-enforced finality, however, SCP manages to do away with those shortcomings of proof of work that render proof-of-work-based systems impractical for payment tokens and redeemable tokens. Therefore, it can be stated that Stellar is highly relevant with regards to tokens, representing the state of the art of blockchain-based systems.

At this point, the reader has been thoroughly introduced into blockchain-based systems and what problems they solve, smart contracts and how they enable tokens, the legal implications of tokens as well as what key properties of modern blockchain-based systems are and how they are achieved. The reader is now able to critically engage in present-day debates on blockchain technology and tokens in conjunction with IoT, CBDC, etc., being able to draw on technical as well as—to a lesser extent—legal knowledge.

References

1. Nakamoto, Satoshi (2008): Bitcoin: A Peer-to-Peer Electronic Cash System https://bitcoin.org/bitcoin.pdf
2. Buterin, Vitalik (2015): A conference talk explaining the Ethereum blockchain protocol https://www.youtube.com/watch?v=gjwr-7PgpN8
3. Buterin, Vitalik (2013): Ethereum white paper https://ethereum.org/whitepaper/
4. Buterin, Vitalik (2017): A tweet considering the Serpent programming language deprecated https://twitter.com/VitalikButerin/status/886400133667201024
5. Bärenfänger, Lucas Konstantin; Ethereum Stack Exchange users (2020): A discussion on the `SplitPot` code example https://ethereum.stackexchange.com/questions/83782/what-are-the-flaws-of-this-example-contract
6. Liechtenstein Government (2019): Report and Application of the Government to the Parliament of the Principality of Liechtenstein Concerning the Creation of a Law on Tokens and TT Service Providers (Tokens and TT Service Provider Act; TVTG) and the Amendment of Other Laws https://www.naegele.law/downloads/2019-07-12_BuA_TVTG_en_full_report.pdf
7. Lokhava, Marta; Losa, Giuliano; Mazières, David; Hoare, Graydon; Barry, Nicolas; Gafni, Eli; Jove, Jonathan; Malinowsky, Rafał; McCaleb, Jed (2019): Fast and secure global payments with Stellar https://www.scs.stanford.edu/~dm/home/papers/lokhava:stellar-core.pdf
8. Mazières, David (2016): The Stellar Consensus Protocol. A Federated Model for Internet-level Consensus https://www.stellar.org/papers/stellar-consensus-protocol
9. Barry, Nicolas; Losa, Giuliano; Mazières, David; McCaleb, Jed; Polu, Stanislas (2018): The Stellar Consensus Protocol (SCP) https://datatracker.ietf.org/doc/draft-mazieres-dinrg-scp/

10. Lokhava, Marta (2019): Intuitive Stellar Consensus Protocol https://www.stellar.org/developers-blog/intuitive-stellar-consensus-protocol?locale=en

11. Chlipala, Jason (2020): Issuer-Enforced Finality Explained https://www.stellar.org/blog/issuer-enforced-finality-explained?locale=en