future internet

Review

Blockchain Technology for Intelligent Environments

Spyros Voulgaris [1,*], Nikos Fotiou [1,*], Vasilios A. Siris [1,*], George C. Polyzos [1,*], Mikael Jaatinen [2,*] and Yannis Oikonomidis [3,*]

1 Department of Informatics, Athens University of Economics and Business, 10434 Athens, Greece
2 Ericsson, 00240 Helsinki, Finland
3 Synelixis Solutions SA, 14343 Athens, Greece
* Correspondence: voulgaris@aueb.gr (S.V.); fotiou@aueb.gr (N.F.); vsiris@aueb.gr (V.A.S.); polyzos@aueb.gr (G.C.P.); mikael.jaatinen@ericsson.com (M.J.); oikonomidis@synelixis.com (Y.O.)

Received: 6 August 2019; Accepted: 3 October 2019; Published: 11 October 2019

Abstract: In the last few years, we have been witnessing the convergence of the physical with the digital world. The Internet of Things (IoT) is progressing at a fast pace, and IoT devices are becoming pervasive in our physical environments, bringing the vision of Intelligent Environments closer to reality. At the same time, the newly-introduced blockchain technology is offering for the first time ever cryptographically proven trust based on a set of mutually untrusted nodes. Blockchain technology thus has the potential to become a key component of many IoT systems, offering them an unprecedented level of accountability, transparency, and reliability. This paper first lays out the principles on which blockchain systems are operating, along with descriptions of the most noteworthy blockchain implementations. It then presents a number of systems through which blockchains may interact with external systems and third-party data sources. Finally, it provides a survey of the state-of-the-art blockchain-based systems targeting IoT applications.

Keywords: blockchains; distributed ledger technology (DLT); internet of things (IoT)

1. Introduction

Intelligent Environments [1,2] are becoming increasingly relevant to our daily lives, as we are witnessing the digital and the physical worlds gradually blending into each other over the last few years. This is evident not only in industrial settings (e.g., smart offices [3], supply-chain provenance tracking [4], industrial automation [5]), but also in leisure activities (e.g., smart homes [6,7], augmented reality games [8]), in healthcare (e.g., autonomous patient monitoring and assisting [9]), in urban planning (e.g., smart cities [10]), and more.

The proliferation of the *Internet of Things* (IoT) [11–13] is a key component of this evolution, as IoT devices are becoming pervasive in our physical environment. Although fundamental technologies that enable this trend, such as diverse types of networking, communication protocols, data analytics, and machine learning algorithms, have received extensive attention by the research community, and major challenges remain unresolved when issues such as *reliability*, *accountability*, *trust*, and *transparency* are at stake.

Blockchains [14,15] constitute a relatively new technology, with no more than a decade of history. Despite their recent conception, blockchains have attracted tremendous attention in a number of diverse fields, most notably in computer science, cryptography, finance, economy, civil law, healthcare, rights management, real estate, auctions, gambling, and generally all industries for which one or more of the aforementioned challenges (*reliability*, *accountability*, *trust*, *transparency*) play key business roles. Some people claim that blockchains will bring to asset management as big of a revolution as the Internet brought to communication.

In particular, for Intelligent Environments and the emerging field of the Internet of Things, blockchains can be a game changer [14]. Through *smart contracts*, blockchains enable for the first time ever provable trust in the interaction between sensors, actuators, and processors that belong to different owners, jurisdictions, and administrative domains. This implies that blockchains can offer a new level of reliability, transparency, and trust to existing or upcoming intelligent environment designs.

This paper explains the principles on which blockchains operate and the most noteworthy implementations of this disruptive new technology, and studies existing blockchain-based systems targeting IoT applications.

More specifically, the paper is organized as follows: Section 2 explains blockchain systems and their inner workings, while Section 3 describes the most notable blockchain implementations. Section 4 presents the challenges of interfacing blockchain smart contracts with the external world and lays out the most notable oracle services deployed to date. Section 5 explores the role of blockchains for intelligent environments and provides an extensive survey of the state-of-the-art blockchain-based systems targeting IoT applications. Finally, the paper concludes in Section 6.

2. Blockchains

In a nutshell, a blockchain is a giant *append-only* log of transactions replicated across a set of participating nodes. When a new transaction is to be appended, participating nodes *vote* on whether it complies with the blockchain's rules and come to an agreement regarding the *admission* and the *order* of new transactions. This agreement is known as *consensus* and the protocol ensuring it is called the *consensus protocol* (of the particular blockchain). Anyone can verify the validity of logged transactions, while no one may tamper with or purge past transactions.

What makes blockchains a disruptive technology is that they offer, for the first time ever, a tamper-proof append-only database where trust emerges through the collaboration of a set of computers, rather than through an institution or organisation that imposes trust from the external world onto the system.

In blockchains, data records (i.e., transactions) are grouped into blocks. The very first block, known as the *genesis block*, is a special block known to everyone. Each other block links to its previous block by incorporating a cryptographic hash (see also Section 2.3.1) of the previous block's contents, creating a *chain* of blocks. In case a block gets tampered with, the hash stored in its successor will not match its (updated) content anymore, effectively breaking the link. That is, tampering with a block deprives it of its successors, rendering the altered block as the last block of this (altered) blockchain. Given the key blockchain policy that, among two or more blockchains rooted at the same genesis block only the longest one is considered valid, tampering with a block will result into a shorter-than-the-original blockchain, which will be instantly ignored by all correctly behaving nodes. That is, tampering with a block is not prohibited by means of traditional authentication and authorization protocols. Instead, it is merely rendered meaningless, as the majority will always follow the longest chain. As creating new blocks is made deliberately hard (see below), an attacker wanting to tamper with a block will have a very hard time in trying to produce a longer chain than the so-far longest one, effectively racing against the rest of the world in terms of computational power.

Blockchains essentially constitute distributed ledgers, which is why blockchain technology is often referred to as *Distributed Ledger Technology*, or *DLT*. Technically speaking, DLT is a wider family of systems, as distributed ledgers that do *not* group records into blocks have been proposed (such as IOTA discussed in Section 3.1.4). However, all dominant distributed ledgers to date are blockchains.

Each blockchain is designed as a combination of certain elements. Identifying and understanding these elements is essential to the comprehension of blockchains and their inner workings, as well as for classifying them into different types. The core elements determining the operation of a blockchain are their *access model*, *token model*, *consensus protocol*, and *smart contract* functionality, as depicted in Figure 1. These core elements are detailed in the following sections.

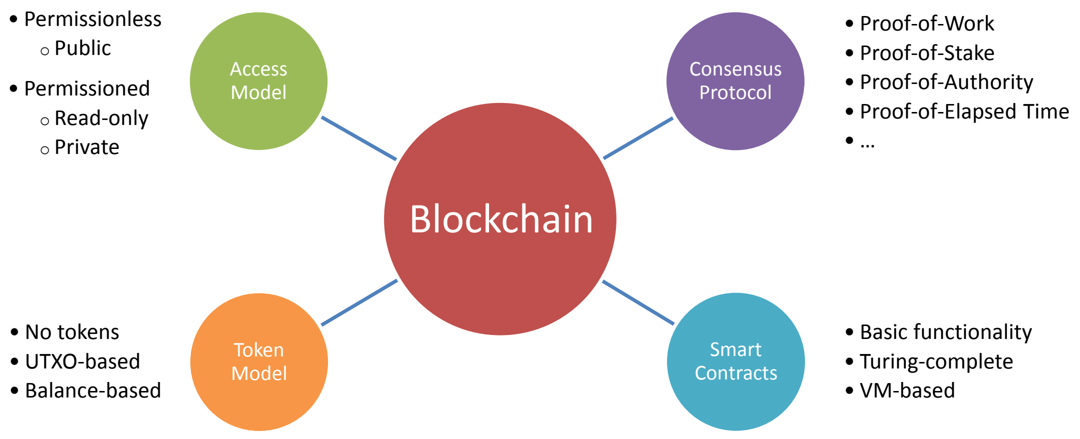

Figure 1. Basic elements of a blockchain system.

2.1. Access Model

The ledgers' policy regarding *which nodes* are allowed to act in *which roles*, places the ledgers into two broad categories: permissioned and permissionless ones.

In *permissionless ledgers*, also known as *open* or *public*, any computer that has network access may join the ledger, basically taking up any role. That is, it may opt to participate as a validator (miner), to contribute in building consensus, as a verifier (full node), to read and locally verify blocks, or simply as a user, to issue new transactions. This model is the most well known one, used in a number of blockchains, including Bitcoin [16], Ethereum [17], Litecoin [18,19], Monero [20], Dash [21], and Dogecoin [22].

In *permissioned ledgers*, a node needs to be authenticated and authorized to take up certain roles. For instance, a ledger could restrict the validators to a predefined set of authorized nodes but let any node locally verify the correctness of the ledger. Other ledgers could also require authentication and authorization to read the ledger, in which case they essentially become *private ledgers*.

There is a considerable variation in the permission model among ledgers. Furthermore, there is an interplay between the permission model and the consensus protocols: the PoA consensus model (Section 2.3.3) is only possible in permissioned ledgers.

2.2. Token Model

Ledgers typically encompass a token model, essentially implementing their own cryptocurrency, used for representing and transferring value within the ledger's realm. For Bitcoin [16], Litecoin [18], and a number of other blockchains, tokens and transactions with tokens have been the exclusive reason of their existence. There are two types of mechanisms to support a token scheme in a blockchain: UTxO-based and balance-based.

UTxO, which stands for *Unspent Transaction Output*, is the token mechanism in Bitcoin. The idea is that each transaction *generates* one or more tokens as its *outputs*, by *spending* some previously-generated tokens, i.e., outputs of past transactions. For instance, if Alice wants to transfer four bitcoins to Bob, while she owns two tokens worth two and three bitcoins, respectively, she may submit to the blockchain a transaction spending her 2-bitcoin and 3-bitcoin tokens, and generating a 4-bitcoin token associated with Bob's wallet, and a new 1-bitcoin token as change, associated with her own wallet. The UTXO model does not store account balances. Instead, a client should trace all spending and generation of tokens for the entire blockchain to locally compute account balances.

The *balance-based* token model is the one employed by Ethereum. It associates a balance with each wallet, which is directly updated by transactions. For example, a transaction in which Alice pays four ethers to Bob simply reduces Alice's balance by 4 and increases Bob's balance by the same amount. Note that, in order to prevent replay attacks (i.e., the resubmission and execution of a past transaction multiple times), a per-user serial number is signed and attached to each transaction, blocking the second execution of any given transaction.

Interestingly, not all blockchains employ a token model. For instance, Hyperledger Fabric [23], discussed in Section 3.2.1, does not include any cryptocurrency. The no-token model can be suitable only for permissioned blockchains, where validator nodes are controlled, authenticated, and typically dedicated to the blockchain's operation; therefore, there is no need for mining incentives and compensation in terms of monetary rewards.

Regardless of whether a blockchain supports tokens or not, and irrespectively of the token model employed, smart contracts can always define and implement their own tokens for use within their code, using any arbitrary token model and rules dictating respective transactions.

2.3. Consensus Protocols

Blockchains' most innovative breakthrough is the creation of trust based on a large number of generally untrusted nodes. This is achieved through sophisticated consensus mechanisms, which, as outlined above, are central to the operation of blockchains. A number of blockchain consensus mechanisms have been devised, having significant differences, yet a common goal: enabling the entire network to decide unanimously and inadvertently on which records to include next, and in which order, into the blockchain. The protocol constitutes, essentially, a *voting mechanism* used for validating and ordering the records that are stored into the blockchain.

In the following sections, we review the most common consensus mechanisms.

2.3.1. Proof-of-Work (PoW)

Proof-of-Work (*PoW*) [16], or *Nakamoto consensus*, is the first and most popular consensus mechanism for blockchains to date, primarily known through its use in Bitcoin. It is based on the combination of (i) the *longest chain wins* principle, (ii) a computationally hard problem to build (or *mine*) the next block, and (iii) a reward for mining a new block into the chain. The most often used computationally hard problem is *reverse hashing*, which is finding a value whose cryptographic hash satisfies some criteria. As cryptographic hash functions are non-reversible, the only way to find a solution is by employing brute-force techniques. Unfortunately, reverse hashing has no real-world scientific or societal causes or benefits beyond just achieving the consensus per se. Its sole purpose is to incentivize participating nodes (known as *miners*) to devote their resources to the chain that has grown to be the longest, i.e., the winning chain, thereby effectively achieving a universal, decentralized consensus on which a specific chain version is unanimously agreed upon.

Unless a single entity controls more than 50% of the world's mining capacity, it is in each miner's interest to abide by the rules. Unfortunately, with the increasing popularity of massive mining pools, the scenario of aggregating more than half of the world's hash power under a single administrative entity is no longer unlikely. Thus, the danger of what is known as the *51% attack* cannot be entirely ruled out.

PoW's main drawback is, however, the exorbitant amount of energy it requires. For example, Bitcoin's mining power is expected to surpass the entire power consumption of countries like Ireland or Denmark by 2020 [24]. This puts the long term viability of PoW-based consensus at stake, notably in terms of their carbon footprint and their effect on global ecology. It has thus fueled strong research efforts in devising alternative consensus mechanisms, some of which are described below.

2.3.2. Proof-of-Stake (PoS)

Proof-of-Stake (*PoS*) [25] is an alternative consensus mechanism, addressing the two main deficiencies of PoW, namely the huge waste of energy and the 51% attack. In order to get the terminology straight, in PoS terms, miners are called *validators* and the act of mining a block is called *minting* or *forging* a block. Contrary to PoW miners, PoS validators are *not* in direct competition against each other in terms of computational power for minting the next block. Instead, the network elects which validator should mint the next block, thus preventing the wasteful process of PoW mining.

The selection of the next block's validator is not random. In order to become a validator, a node has to deposit something (essentially a pledge or collateral of something considered valuable by the community) as a security deposit, known as its *stake*. This stake remains reserved and is returned to its owner, along with all the transaction fees of the minted block, only after some time has passed. In case the validator has approved a fraudulent transaction, it loses all transaction fees *and* its originally pledged stake. This creates a profit-based motivation to follow the protocol.

The rationale behind electing a validator proportionally to its stake is straightforward: The more you risk losing, the less likely you are to cheat. This, however, encourages a rich-gets-richer pattern, where the richest nodes are repetitively appointed validators, collecting all transaction fees, and getting even richer. In order to alleviate this problem, the use of *coin age* has been devised: A coin's power in increasing a node's chances to be appointed a validator is dependent on the time since the coin was last used as stake. That is, coins that were recently used as a stake are of no use in being used again, until a certain time period has passed.

Although PoS significantly lowers its ecological impact as a consensus algorithm compared to PoW, it is not without drawbacks. The fact that PoS operation costs are negligible enables a single validator to participate in multiple PoS blockchains using the same machine. In particular, when a blockchain is forked into many concurrent versions, nothing prevents a validator from supporting any number or even all of them simultaneously, as its profits will increase regardless of which fork becomes dominant. As a result, forks on PoS chains may take an arbitrarily long time to settle, which diminishes the blockchain's append-only, tamper-proof nature. This constitutes a new attack vector known as *Nothing at Stake*. In contrast, when a fork occurs in a PoW blockchain, miners are indirectly forced to choose one fork to continue supporting due to the prohibitive computational capacity required.

Additionally, the *Tragedy of the Commons* also facilitates the above modus operandi. If the stake of a miner who wishes to initiate a fork in a PoS system is sufficiently large, other miners will join his fork in fear of losing their stake and being unable to overcome and prevent the fork.

As this problem exists inherently in the absence of a restrictive resource for operating nodes in a PoS scheme, its solution remains the subject of academic research initiatives. One such initiative is called *Ouroboros*, led by the Universities of Edinburgh, Connecticut, and Aarhus, and a private organisation named IOHK. They explore a solution based on the forking mechanism of a blockchain.

There are a number of systems using various flavours of PoS, including Cardano [26], Lisk [27], BlackCoin [28], Peercoin [29], and Nxt [30] (some of them discussed later in this paper). Ethereum has announced the intention to switch to PoS.

2.3.3. Proof-of-Authority (PoA)

Proof-of-Authority (*PoA*) [31,32] delegates transaction validation and block creation to a certain set of *authorized nodes* who are acting as the administrators of the system. This paradigm best fits permissioned blockchains, as it exhibits a centralized operation: denoting who are eligible as administrators (and who are not). Transaction throughput can be high and blockchain parameters can be fine-tuned to the specific needs of the private networks they serve. However, trust does not emerge from the inherent dynamics of stakeholders but is rather *outsourced* to the sysadmins, guaranteeing the secure and flawless operation of a big enough fraction of the authorized nodes.

Most of the PoA-based consensus mechanisms are based on the so-called Byzantine consensus protocols. These protocols stem from the seminal work of Lamport, Shostak, and Pease in the early 1980s [33,34] and later works of Castro and Liskov [35]. While many Byzantine protocols are being employed elsewhere, e.g., in many Cloud databases, their use in blockchains is mainly differentiated through the blockchains being append-only databases.

2.3.4. Proof-of-Elapsed Time (PoET)

Proof or Elapsed Time (*PoET*) [36] is a consensus mechanism introduced by Intel (Santa Clara, CA, USA), making use of their CPUs' *Software Guard Extensions* (*SGX*) feature, enabling processors to run trusted code that cannot be tampered with. The logic behind PoET is simple. Each participating node waits for a *random period*, and the first node to finish waiting becomes the validator for the next block. Essentially, this is a basic leader election protocol, which uses negligible CPU power to execute. Clearly, its correctness directly depends on nodes being honest with respect to waiting for a *really randomly chosen* time period. However, this is guaranteed through code that is trusted based on Intel's SGX feature.

One example of use of PoET is in Hyperledger Sawtooth [37], developed primarily by Intel.

2.4. Smart Contracts

Smart Contracts bring another groundbreaking innovation to the blockchain world. Rather than using blockchains' decentralized trust model for offering just an immutable decentralized append-only data store, they exploit the mechanisms to provide a tamper-proof decentralized *world computer*. Through smart contracts, blockchains are promoted from special-purpose tools serving a single application (e.g., Bitcoin) to general-purpose platforms, allowing developers to deploy and execute custom code that may implement arbitrary application logic.

A smart contract is essentially a program running on a ledger, maintaining some internal state, and updating this state through transactions. To allow transactions to be made, a smart contract exports an API consisting of one or more functions. Read-only functions, i.e., functions that only read the smart contract's program state without changing it, are executed locally on the node that calls them. Calling such functions does not create new transactions to the underlying ledger. Read-write functions, i.e., functions that update the smart contract's state, are executed on all nodes running the ledger, and require new transactions to be entered into the ledger. More specifically, all nodes that validate a block now, or any time into the future, have to execute all smart contract function calls included in that block, in order to check whether the resulting overall state matches the state recorded into the block's hash.

Even more specifically, smart contracts gain their power through the following fundamental features:

- **State storage:** It is possible for a smart contract to store its own state into a ledger. This makes it possible for a smart contract to store arbitrary data, not just data specific to Bitcoin or some other application.
- **Turing-complete computation:** Many smart contracts enjoy a Turing-complete computation environment. This allows them to implement arbitrary logic for any type of custom ledger applications.
- **Interaction with other contracts:** A smart contract can call functions of other contracts to interact with them.
- **Input from the external world:** A smart contract can get input from external sources, such as what is the current cryptocurrency value in fiat money (euro, dollar, etc.). This can be done indirectly through interaction with other contracts, which, in turn, are updated by (trusted) external organizations pushing frequent updates to them. These, so-called, *oracles* are discussed further in Section 4.
- **Input from the ledger itself:** A smart contract is generally allowed to read and use any value in the ledger, such as the last block's hash value. For example, the block hash value can serve as a deterministic pseudo-random number generator.

A platform supporting smart contracts essentially allows arbitrary applications to be developed, and to benefit from the same level of trust and transaction irreversibility enforced by a large community of participating nodes.

The possibilities opening up with smart contracts are unlimited. Some examples can help to better understand possible scenarios:

- **Multisignature wallets:** A group of two or more people may set up a shared wallet, configuring it in such a way that only the *combination* of all signatures allows the withdrawal of arbitrary amounts, but a single signature *alone* may spend e.g., up to 1% of the stored amount on a daily basis.
- **Insurance:** A company could issue insurance through smart contracts, e.g., letting a farmer insure his crop against bad weather conditions. The smart contract would require the farmer to pay his insurance fees up to a given deadline, and would cover that farmer's potential loss based on input from a trusted weather-logging smart contract, which, in turn, would get its input from an externally trusted IoT system (an *oracle*).
- **Auctions:** People could bid on an asset in a system without a middle man, in such a way that, for every bid, it is verified that the owner indeed has that amount, and by reserving that amount until either a higher bid is made or a timeout elapses and the purchase is completed.
- **Prediction markets or betting:** Bets could be set on predicting future events that can be verified through trusted feeds (e.g., which team will win a soccer tournament).
- **Event driven communication:** Some smart contract platforms support *events*. These events are generated by the smart contracts and are propagated to all nodes of the blockchain network. Applications may *register* their interest in some specific events and get notified when these events take place.
- **Access control:** Smart contracts can be used for linking payments to authorization grants, as well as for immutably recording authorization information and policies in the blockchain. For example, Siris et al. [38] enhance OAuth 2.0-based authorization for IoT devices using smart contracts.

3. Noteworthy Blockchain Implementations

The number of reported cryptocurrencies has exceeded the whopping number of 2000, as of September 2019. Clearly, a large fraction of them are identical to each other from a technical point of view. Hence, there would be no point in even listing them all. Instead of that, we collected a number of well-known (but different) ledgers, i.e., ones that may be considered noteworthy for their technical implementation.

In this section, we present a number of the most notable blockchain implementations, and elaborate on their operation.

3.1. Public/Permissionless Blockchains

3.1.1. Bitcoin

Introduced by the pseudonymous Satoshi Nakamoto in [16], Bitcoin undoubtedly earns first place in this list, being the seminal blockchain design and implementation, as well as the first successful fully decentralized digital currency.

Bitcoin's blockchain is used exclusively for transactions in its own cryptocurrency (called *bitcoin*). That is, there is no support for smart contracts or arbitrary applications. Bitcoin does support some elementary scripting; however, the sole purpose of the scripting is limited to providing flexibility in setting the conditions for spending assets, e.g., requiring either a single user's key, or *n* users' keys, or *k*-out-of-*n* users' keys, etc., in order to dispose of assets.

Bitcoin is an open-source protocol running on its permissionless blockchain. It employs a Proof-of-Work consensus mechanism, based on double SHA-256 reverse hashing. As an explicit design decision, Bitcoin generates one block per 10 min, on average. This is achieved through a *difficulty* parameter for the Proof-of-Work algorithm, which is automatically adjusted every 2600 blocks to counterbalance global hashrate increase (or decrease), by considering how much faster (or slower) than 10 min the average generation of the last 2600 blocks (about 18 days) was. The incentivization mechanism to attract miners consists of allowing them to deposit to their wallets a predefined amount of newly generated bitcoins for each block they mine. This is also the only way bitcoins are being generated globally.

There are significant scalability concerns regarding Bitcoin, as its transaction processing rate of ca. 5–7 transactions per second is deemed too low for serving trade at a global scale.

3.1.2. Ethereum

Ethereum [17] was the first (and is to date the largest) deployed blockchain to support smart contracts. It was proposed by Vitalik Buterin in 2013 in the respective *Ethereum White Paper* [39], and elaborated in far more detail by Galvin Wood in 2014 in the so-called *Ethereum Yellow Paper* [40]. It was then deployed in July 2015. It was the first widely-deployed blockchain to incorporate the notion of smart contracts, explained in Section 2.4 above.

Ethereum comes with its own currency, called *ether*. Its use for Ethereum's operation is fundamental in two ways. First, it constitutes the incentive for *validators* (Ethereum term for *miners*) to contribute their resources to the Proof-of-Work algorithm. Second, it is used to regulate the use of the blockchain's resources by charging for their use. More specifically, ether is needed to pay for *gas*, a unit used to measure the computation, storage, and bandwidth cost an operation imposes on the blockchain. In order to invoke a smart contract function, the caller has to specify how much ether he is paying per gas spent, as well as an upper limit on the gas that can be spent. This way, Ethereum can support a Turing-complete virtual machine, called the *Ethereum Virtual Machine* (*EVM*), without fearing a denial-of-service abuse of the system: any long- or eternal-running function costs money, and will eventually be killed for having run out of gas.

A high-level, Object-Oriented Programming (OOP) language for smart contracts, Solidity, has been defined in order to enhance (and target) the EVM and the process of verifying and enforcing constraints at compile-time rather than run-time. It is statically-typed and has similar syntax to ECMAScript (JavaScript). As an OOP language, it supports composition, inheritance (single and multi-level), encapsulation, polymorphism, interfaces, and abstraction. Smart contracts in Solidity are similar to classes in OOP and the curly-braces control structures are also available (e.g., if, while, etc.).

In Ethereum, it has been a design decision to produce a new block every 14–15 seconds on average, with clear user experience benefits stemming from much faster transactions than in Bitcoin. At this block generation rate, there is a non-negligible probability for more than one block to be mined in parallel, effectively creating a *fork* until subsequent blocks determine the *winning block* and the *orphan block(s)* among the competing blocks. Additionally, as 14–15 seconds are comparable to the time it takes to disseminate a block to the entire network, receiving a new block a few seconds earlier gives a significant advantage, which would normally favor large pools of nodes that could give higher priority to disseminating new blocks among themselves before spreading them to the rest of the community. In order to weaken this effect, Ethereum does not only reward blocks that do get accepted into the main chain, but also (valid) orphan/stale blocks that were on the abandoned path of a fork. For this reward to be handed out, a newer block (called *nephew*) should link to a past stale block (called *uncle*) in addition to its direct parent. Then, the uncle block receives 87.5% of a new block reward, and the nephew receives the remaining 12.5% of that reward, as an incentive to include it.

3.1.3. Cardano

Cardano [26] is being developed in two layers that separate the ledger of account values from the reason why values are moved from one account to the other. The Cardano Settlement Layer (CSL) acts as the *balance ledger*. It uses a Proof-of-Stake consensus algorithm to confirm transactions and generate new blocks. Moreover, CSL supports sidechains for moving assets from the CSL to the Cardano Computation Layer (CCL) and any other blockchains that support the Cardano KMZ protocol, which is used for efficient Simplified Payment Verification (SPV) proofs [41]. CCL is the second layer of the

Cardano platform. It contains the information on why transactions occur. Because the computation layer is detached from the CSL, different users of the CCL can create different rules when evaluating transactions. The Cardano team is creating a new programming language to be used to develop smart contracts on the CCL, which is called Plutus. CCL also supports Solidity—used in Ethereum smart contracts—for low-assurance applications.

The Cardano ledger will also implement its own proprietary virtual machine, called *IELE* (named after a mythical creature). The VM is based on the LLVM compiler (originated in the Low Level Virtual Machine project) [42], along with its own unique low-level language. The core difference between IELE and the Ethereum VM is that IELE is a register-based machine with an unbounded number of registers, while EVM is a stack-based machine with a stack limit of 1024. As direct result of IELE's design paradigm, it supports unbounded integers, enabling easier development of secure smart contracts. The rationale behind the development of IELE is the creation of a uniform low-level language that elegantly translates to and from high-level languages, due to its register-based nature.

3.1.4. IOTA

IOTA is a rather different type of a distributed ledger, in the sense that it is *not* based on a blockchain structure. Instead of letting miners confirm transactions in blocks, IOTA decentralizes this process even further, requiring users themselves to confirm each other's transactions. More specifically, each new IOTA transaction has to include *links* to at least two past transactions, which it *approves directly*. By linking to them, it also *approves indirectly* all past transactions approved (directly or indirectly) by them, all the way to IOTA's genesis transaction. This creates a directed acyclic graph (DAG) linking newer transactions to older ones. Contrary to other distributed ledgers, this process does not involve any transaction fees, as the approval of the transactions is made by the users themselves.

Attempting to approve a transaction that is either invalid in itself, or which has approved (directly or indirectly) some invalid transaction, will result in getting the transaction eventually being neglected, and therefore dropped. This constitutes a strong motivation for performing a thorough check when approving past transactions, essentially delegating the task of banning invalid transactions and maintaining consensus to the users themselves. The more new transactions approve (directly or indirectly) a given transaction T, the higher is the confidence on T being valid. There is no fixed confidence level for considering a transaction definite; it is subject to the risk level each user deems acceptable.

Currently, the users are not allowed to pick arbitrary transactions to approve. Instead, a centralized third party, known as the *coordinator*, allocates approval tasks to users submitting new transactions. This is considered a temporary measure that is planned to be dropped later on.

A core advantage of IOTA concerns its scalability. As registering a transaction requires checking and approving two other transactions, the IOTA system enjoys an inherent elasticity: the higher the transaction rate, the higher the collective capacity of *approvers*. An important issue remains: to find the right motivation for ensuring that all pending transactions will be picked for approval, without any of them being left waiting indefinitely. Another unique feature of IOTA is the use of *Winternitz One-Time Signatures*, which are safe even for post-quantum usage. On the other hand, the use of this type of signature results in big transactions, in the order of 10 KB, far larger than Bitcoin's transactions that have an average size of 600 bytes.

IOTA is still in a beta phase and has received some negative criticism; for example, MIT's Media Lab criticizes it for incorporating custom cryptographic solutions [43,44].

3.2. Private/Permissioned Blockchains

3.2.1. Hyperledger Fabric

Hyperledger Fabric [23] is one of the first completely permissioned distributed ledger implementations that is designed for enterprise usage. Following the norm of the latest released ledgers, it is a smart contract capable ledger in which a contract is referred to as *chaincode*. The core feature of Hyperledger Fabric is the modularity its internal operational structure provides. The various services that contribute to its operation are split into secluded containers that are independent of each other, enabling users to hot-swap integral components of the system, such as the consensus mechanism or the virtual environment enabling the execution of chaincode, with new and completely different ones.

This novel modularity enables Fabric to facilitate the operation of chaincode within conventional operating systems. This, in turn, allows developers to create chaincode in general-purpose programming languages, such as Python and Java, with Go being the language of choice for Fabric. The only prerequisite of a fully functional chaincode is the deterministic execution of its functions, as well as the conformity of its specification to the Chaincode Interface, as defined by Hyperledger Fabric.

Being a permissioned ledger, Fabric employs a layer of membership delegation through the usage of a *Public Key Infrastructure*. Each entity within the Hyperledger Fabric network is uniquely identified by its cryptographic key-pair, for which a valid certificate is assigned by the *Certificate Authority* (*CA*) operating in the network. Although CAs are part of the Fabric network, the ledger operators are split into two subcategories, the *Orderers* and the *Peers*. The Orderers are in charge of, as their name implies, ordering the transactions within the blockchain network, resulting in an atomic broadcast of new blocks. Orderers implement the consensus mechanism of the ledger and ensure that the ledger is kept in synchrony among all the Peers. The Peers, on the other hand, simply act as hosts of the ledger and are in charge of maintaining the ledger network by keeping a copy of the ledger and allowing clients to interact with it.

The Hyperledger Fabric network does not contain any form of a cryptocurrency. Instead, all transactions are treated as chaincode invocations within the network. Whether a chaincode invocation is valid and should be included in a new block is defined by the chaincode itself once it is deployed on the network. Specifically, a chaincode may specify a set of signatures that need to be acquired by Peers operating within the network in order for a new transaction to be submitted on the blockchain that will alter the chaincode's state. The Peers that are able to endorse chaincode transactions are called *Endorser Peers* and they provide their endorsement upon successful execution of the chaincode within their respective environment.

The definition of a network in Hyperledger Fabric differs from many other ledgers in that it enables multiple ledgers to operate within the same network, via the segmentation of the network into different *channels*. Each channel retains a separate distributed ledger, which can communicate with the ledgers running on other channels within the same network. This feature enables organisations that operate on the same network to keep private information within their own peers, by maintaining their own distributed ledger.

Hyperledger Fabric is part of the Hyperledger project, an umbrella project of open source blockchains and related tools of the Linux Foundation supported by key industry players such as IBM, Intel, and SAP. Under this umbrella, there is also Hyperledger Indy, discussed next, Hyperledger Sawtooth, mentioned later, and other projects and tools.

3.2.2. Hyperledger Indy

Hyperledger Indy [45] is a permissioned ledger used for managing *Decentralized Identifiers* (*DIDs*) [46]. It originated from the Sovrin project [47] and it is currently under incubation by the Hyperledger foundation.

DIDs is an identity technology that tries to overcome the limitations of centralized identity systems, including the federated ones. A key aspect of DIDs is that they are designed not to be dependent on a central issuing party that creates and controls the identity. Instead, DIDs are managed by the identity owner, an approach known as *self-sovereign identity*. The W3C Credentials Community Group has published a DIDs specification document, targeting the interoperability among various DID systems. The specification defines a DID as a random string associated with a *DID Document* that includes public information about the DID owner and which is stored in a ledger, as well as with secret information used for the DID verification, stored in the DID owner's wallet. Furthermore, DIDs can be associated with some attributes referred to as *claims* or *verifiable credentials* (VCs). VC owners can securely prove to a third party that they are the legitimate owner of a specific credential (e.g., being an adult in a given jurisdiction, without necessarily revealing their identity).

Hyperledger Indy abides by the W3C Credentials Community Group specification. It implements a permissioned ledger for storing DID documents, using the *Plenum* consensus protocol, i.e., an implementation of the Redundant Byzantine Fault Tolerance algorithm [48] that can sustain F failures, where $3F + 1$ nodes are present. Furthermore, Indy provides a convenient SDK that can be used for creating, managing, and using digital identities. An Indy ledger is managed by a pool of *Stewards*. Stewards are in essence the entities that validate identifiers before they are recorded. Identifiers are generated by users and are forwarded to a Steward through a *Trust Anchor*, a specialized network node that has a trust relationship with a Steward and that vouches for the validity of an identifier.

3.2.3. MultiChain

MultiChain [49] is a platform for the creation and deployment of private blockchains. It supports both intra- and inter-organizational deployment and it offers managed permissions, rapid deployment, unlimited assets, data streams, customizations, and bitcoin compatibility. It is derived from the Bitcoin Core software, but it is not limited to the Bitcoin blockchain. MultiChain aims to provide privacy and control in an easy-to-use package so that the deployment of blockchain technology will no longer be a key obstacle. At the same time, advantages of it being a private blockchain are eminent: (i) no scalability issues, and (ii) the blockchain only contains transactions relevant to the participants.

MultiChain applies integrated management of user permissions to solve mining, privacy, and openness problems. By using public-key cryptography, which enables any message to be signed by a user and not just transactions, the platform is able to restrict blockchain access to a list of permitted users, embedding a permission protocol in the handshaking process that occurs during the connection of two blockchain nodes. The same principle can also be extended to other operations, such as the right to send and/or receive transactions to a set of addresses. In that way, the MultiChain platform manages to:

- Ensure that blockchain activity is only visible to selected participants,
- Introduce controls and permissions on transactions,
- Enable mining to take place free of Proof-of-Work and the associated costs in a secure way.

The risk of a single participant of a private blockchain to monopolize the mining process is mitigated by placing a constraint on the number of blocks a single miner may create within a given window. In addition, as mentioned above, MultiChain is not limited to a specific blockchain, as it is easy to configure (configuration during uptime is reported as future work) and deploy, not just by specialized developers, but also by system administrators. MultiChain also supports multicurrency, utilizing the same techniques used in CoinSpark [50], and moves even further by improving on these schemes by integrating support for third party assets directly into the chain's rules.

A main focus of the MultiChain platform is to provide a way for smooth transitions between private blockchains and the bitcoin blockchain in either direction. This is achieved by the following:

- MultiChain is based on a fork of Bitcoin Core, its interface is fully compatible with it, and it can act as a node on the regular bitcoin network.
- It utilizes Bitcoin's protocol, transaction, and blockchain architecture, enhancing just the handshaking process.
- It adopts multicurrency and messaging features offered also by the CoinSpark protocol, to enhance bitcoin transactions and to allow applications which use asset tokenization and messaging to move between Bitcoin and private blockchains with minimal code changes.

While there does not appear to be any published criticism specifically on MultiChain, there appear to be at least two problematic areas:

- Any blockchain that uses Bitcoin's protocol and block structure are open to 51% attacks by existing Bitcoin miners [51].
- Using PoW in a private blockchain is unnecessarily energy inefficient, since, in private blockchains, PoA or PoS can be used.

3.2.4. Corda

Corda [52] is a blockchain-inspired distributed ledger by R3, a financial technological company based in New York. Corda is meant to serve the purpose of exchanging information between nodes belonging to different companies, to record, manage, and synchronize agreements and transfer value. It is designed to be fast and easy to adapt for building customer specific solutions.

Corda relies on various existing technologies including Java and enterprise technologies, relational databases and the Advanced Message Queuing Protocol (AMQP). It does not have any internal Corda coin. Corda focuses on the critical issues of privacy, transaction finality, legally identifiable participants, scaling, developer productivity, and easy enterprise integration.

Corda goals include: assurance that the parties involved in the same transaction agree on the facts, a mechanism for reaching consensus on the state of the ledger, and immutability of the ledger and transactions. Corda is based on: a point-to-point architecture, a pluggable consensus, and a multilateral ledger.

Corda transactions are similar to those of Bitcoin: they have inputs, outputs and signatures, but, unlike Bitcoin, they can contain arbitrary data [53]. Corda does not order transactions using a blockchain and does not use miners or PoW. Instead, each state points to a notary, which is a service that guarantees it will sign a transaction only if all the input states are un-consumed. It is a permissioned DLT and identities are used throughout and at all levels. No broadcast is used. This identity does not have to be a legal or true identity.

A Corda network has one or more notary services which provide transaction ordering and timestamping. Notaries are typically composed of multiple mutually distrusting parties who use a standard consensus algorithm. Notaries consider transactions submitted to them and either sign or reject them. The presence of an appropriate notary signature indicates transaction finality.

4. Oracles: Interfacing with External Systems

Distributed ledgers are deterministic systems: the outcome of a series of smart contract computations is expected to be always the same, regardless of *which* ledger node performs them and *when* in the future. Determinism is necessary for the ledger consensus mechanisms to work. Contrary to the determinism of a ledger, the real world is (apparently) non-deterministic. Hence, directly considering real-world data (e.g., web content or external documents) in a smart contract would break the blockchain consensus process, as distinct nodes in the ledger could perceive differing values. Hence, distributed ledgers do *not* allow smart contracts to access information outside the ledger.

Oracles help circumvent this limitation by obtaining data from *trusted sources* outside the ledger, and making it available for use within the ledger. Oracles are smart contracts in themselves, which allow only selected trusted software modules that are external to the ledger to alter their state and to update their data. Hence, they can be seen as bridges interconnecting the ledger with the outside world.

Oracles enable two-way interaction between a ledger and the outside world, resolving real world details that cannot be known at the time a smart contract is written or published on the ledger [54]. In the remainder of this section, we consider some of the best known oracles.

Town Crier [55] acts as a bridge between smart contracts and existing websites, which are already commonly trusted for non-blockchain applications. It combines a blockchain front-end with a trusted hardware back-end to scrape HTTPS-enabled websites and serves source-authenticated data to relying smart contracts. Town Crier also supports confidentiality. It enables private data requests with encrypted parameters. Additionally, in a generalization that executes smart contract logic within Town Crier, the system permits secure use of user credentials to scrape access-controlled online data sources.

Oraclize [56] allows smart contract developers to fetch the results of arbitrary computations performed on systems, such as on an AWS virtual machine. Specifically, developers can upload a zip archive containing a Dockerfile, along with any necessary data, to Inter-Planetary File System (IPFS) [57]. Then, developers can include within their smart contacts queries to an Oraclize smart contract that contains the IPFS hash of the archive. An Oraclize module monitors the ledger and, once it identifies that the Oraclize smart contract is queried, it triggers an Oraclize AWS instance to initialize and execute the Docker application identified by the IPFS hash provided by the developer's smart contract. Once the computation is completed, the Oraclize module performs a smart contract call that sends the result of the computation back to the original smart contract. An interesting extension supported by Oraclize is that it can use a *TLS notary proof* [58], which can be used to check the integrity of the response provided by Oraclize.

ChainLink [59] provides a decentralized oracle network, which includes on-chain components for smart contracts to request external connectivity and an off-chain consensus mechanism, reputation, and bidding system to ensure that the external data are trustworthy, even though it is provided by independent oracle service providers. With ChainLink, smart contract developers do not contact oracles directly, but rather submit service requests, which include requirements in the form of SLAs, to an order-matching contract. Oracle providers can monitor and filter requests based on their capabilities and service objectives. Once an SLA request has received a sufficient number of qualifying bids, the set of oracles is selected from the pool of bids. Similarly to other approaches, ChainLink tokens provide the economic incentives for oracle providers to be truthful. While ChainLink focuses on the Ethereum blockchain, Hivemind [60] is another decentralized oracle proposal that is developed focusing on Bitcoin. Similarly to other proposals, Hivemind uses economic incentives and reputation to ensure trustworthy reporting of external data.

The solutions presented above are data oracles [61]: they read data from external data sources; therefore, these data sources need to be trusted. Computation oracles go one step further and have the oracle nodes actually perform the relevant computations. One proposal of a computation oracle was the SchellingCoin protocol proposal. It proposed a decentralized network of oracles that would perform computation, providing incentives by rewarding participants who submit results that are closest to the median of all submitted results. The proposal included a verification model that was based on m-out-of-n oracles performing computation and voting on the correct result; each node had the ability to challenge results by submitting a security deposit.

Another proposal for supporting verifiable computation is *TrueBit* [62], which introduces a system of solvers and verifiers. *Solvers* are compensated for performing computation and *verifiers* are compensated for detecting errors in solutions submitted by solvers. In the event of a challenged solution, solvers and verifiers play an interactive verification game such that only a small portion of the computation is performed on-chain after a number of rounds during which the challenger disputes increasingly smaller subsets of the original computation.

Microsoft's *Coco Framework* is a project under development that targets leveraging existing blockchain technologies to support business processes [63]. In particular, Coco's goals are to deliver better performance, business logic confidentiality models, distributed governance, and reduced energy consumption to existing blockchain protocols, including Ethereum, Quorum, Corda, and Hyperledger Sawtooth. Coco promises to fulfil these goals through the use of trusted execution environments, such as Intel's SGX and Windows Virtual Secure Mode, enabling the creation of a trusted network of physical nodes on which to run a distributed ledger.

5. Blockchains for IoT Systems

The rapid emergence of IoT frameworks and ecosystems underlines the need for higher *reliability*, *accountability*, *trust*, and *transparency*. This calls for a fusion between IoT ecosystems and blockchain technology, a trend that is swiftly gaining momentum. To develop a better understanding of how these technologies blend together and the respective benefits, we present two application scenarios from the real world, namely a *food supply-chain* and an *energy marketplace*.

Application Scenario 1—Food Supply-Chain: In a typical food supply-chain, produce is grown in farming fields, it is packed and transferred via land transportation, it may be loaded on cargo ships, it is stored in large warehouse depots, and it eventually reaches supermarket shelves to be picked by consumers, as depicted in Figure 2. The entire transportation process needs to be carried out in compliance with strict rules and conditions to meet certain quality standards so that produce reaches consumers in a fresh and well-preserved state. When, however, this fails to happen, it is far from trivial to safely identify the link in the chain where conditions were violated, given the number of different parties involved.

In order to alleviate this issue and to enhance accountability, IoT solutions are deployed across the supply-chain, to record and store sensor and quality assurance data. Such data include temperature, humidity, wind, irrigation, fertilizers used, produce weight, produce ripeness level, etc., at various stages of the supply-chain. However, all these data are in principle stored in proprietary data stores, private silos of the companies involved. Thus, in case of dispute, it is hard to verify the validity of proprietary data presented by conflicting-interest parties.

Blockchains offer the transparency and trust needed to provide a dependable accountability basis. Data produced at different stages of the supply-chain can be reliably stored in a blockchain managed by all participating parties together, being a solid reference to ground truth in case of disputes. Figure 2 illustrates a high-level abstraction of a food supply-chain.

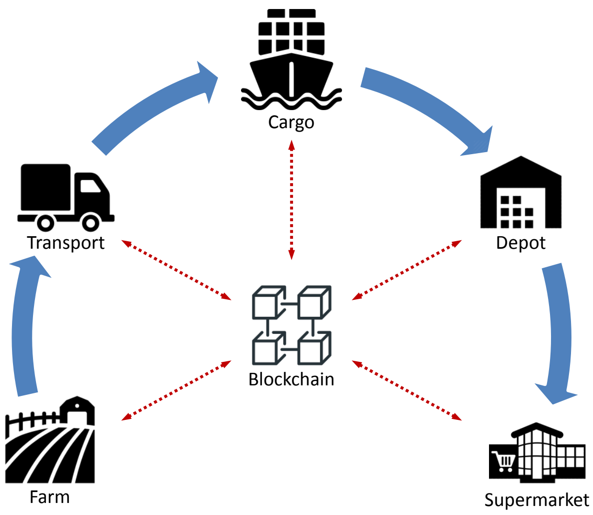

Figure 2. Food Supply-Chain: Data gathered from all stages of a food supply-chain are stored in a blockchain to provide dependable accountability with respect to proper handling of produce across the whole chain.

Application Scenario 2—Energy Marketplace: In the last few years, renewable energy sources, such as solar panels and wind turbines, are becoming more popular and widespread, shifting energy production from a traditionally centralized process to a decentralized paradigm. Meanwhile, electric vehicles are gaining an increasingly wider slice of the worldwide market, becoming the first ever power-hungry devices that are inherently mobile, hence flexible with respect to selecting their charging spot. Finally, research on high-capacity batteries and energy storage systems is advancing, making it possible for individual households to play the role of small energy aggregators.

The development of these new technologies creates the need for an efficient producer/consumer energy marketplace. In a free market, each producer's selling price will fluctuate in real time to reflect their current power availability. Wind turbines will produce cheaper electricity on windy days, just like solar panels on sunny days. Owners of energy storage systems will be lowering their prices when in power abundance, while they will tend to charge higher rates when low on energy reserves. Electric vehicle owners will also experience diverse urgency in charging their batteries, depending on energy level, energy prices, current location, planned route, etc.

This logic may be coded in a blockchain smart contract, bringing in the transparency and reliability required in such a dynamic, complex, and multivariate energy marketplace. In such a system, vehicle owners, energy producers, and energy aggregators will be trading energy at a fine granularity, as illustrated in Figure 3.

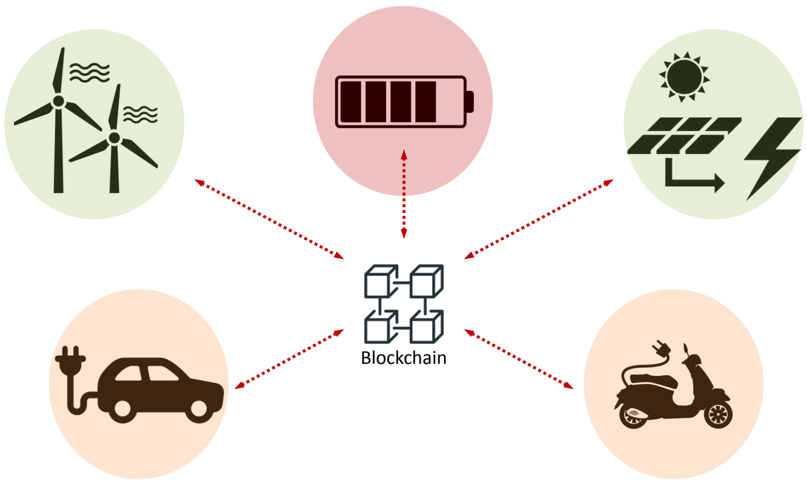

Figure 3. Energy Marketplace: Energy producers, energy aggregators, and electric vehicles mingle in a real-time, fine-granularity trading, to improve the fair distribution of energy and to prevent energy shortages.

In response to the above, a number of IoT frameworks have embraced blockchain technology into their platforms. At a high level, these efforts may be split into four broad categories. The first category concerns *generic, all-in-one systems* developed by organizations established in the IoT world that attempt to combine in-house IoT solutions with private blockchains, typically for industrial use. The second category consists of community-developed blockchain solutions that target *data delivery and storage*, offering a blockchain-based messaging and storage middleware for fast and secure interaction between IoT devices. The third category comprises *marketplace-oriented systems*, aiming at facilitating the creation of data marketplaces and trading therein. Finally, the fourth category consists of blockchains that serve as *timestamping services*, allowing clients to *timestamp* arbitrary data, so they can prove anytime in the future that they were in possession of that data at a specific moment in time.

In the following sections, we visit a number of representatives of these broad categories, discussing their specific design and features.

5.1. Generic, All-In-One Systems

5.1.1. Hyundai Digital Asset Currency

Hyundai Digital Asset Currency (*HDAC*) is a platform that targets to combine IoT technologies with a blockchain-based solution in order to provide *authentication*, *mapping*, and *machine-to-machine* (*M2M*) communication [64]. The HDAC platform implements a private blockchain that provides a simple transaction environment targeting M2M communications. HDAC plans to provide a hardware-based wallet that will enable users to interact with its blockchain, as well as a public blockchain that will be used for inter-connecting private ones.

HDAC is based on MultiChain. It targets 160 transactions/sec in the public blockchain and 1000 transactions/sec in private blockchains. It implements a consensus algorithm based on Proof-of-Work (PoW), code-named ePoW, targeting energy efficiency. The distinguishing characteristic of ePoW is that it limits the number of miners participating in a particular mining task by specifying a *block window*, a period of time during which miners are not allowed to mine blocks.

The main component of the HDAC platform is the *IoT contract*, a smart contract associated with IoT devices to interact with the blockchain. Using IoT contracts, Thing owners can specify operations for their devices to perform, as well as security and access-control policies.

5.1.2. IBM Watson with Hyperledger Integration

IBM offers *Watson IoT with Blockchain*, a system enabling IoT devices to send data to private blockchain ledgers for inclusion in shared transactions with tamper-resistant records. IBM Blockchain on Bluemix (IBM's Cloud platform) is the commercial version of Hyperledger Fabric, of which IBM is a premier member.

IBM promotes its blockchain platform for use in a number of industries, including banking and financial markets, insurance, retail and consumer goods, government, healthcare, automotive, travel and transportation, and media and entertainment. It remains somewhat unclear how IBM plans to apply their blockchain architecture in the IoT domain.

5.1.3. Samsung Nexledger

Samsung Nexledger is a permissioned blockchain platform targeting enterprises in different industry domains. Nexledger claims to already support the following use cases:

- **Digital Identity:** Solution that utilizes blockchain technology to create digital identities for customers.
- **Digital Payment:** Blockchain-integrated solution that can supplement credit and debit cards for customer payments.
- **Digital Stamping:** Blockchain-based solution that can be used for creating secure digitally stamped signatures without the need for a third-party authenticator.

Support for new use cases can be added on as applications on the platform. Nexledger claims support for the following features in their solution:

- Bolstered contingent measures with management monitoring of block information.
- Reduced lead time with improvements in transaction verification and processing algorithm.
- Reduced resource consumption with improvements in the confirmation racing algorithm for PoW.
- Optimized management for the distributed ledger with multi-chains and partitioned chains.
- Enhanced security with FIDO (Alliance https://fidoalliance.org/) certifications and multiple biometric modalities.

5.1.4. AWS Blockchain Partners Portal

Amazon Web Service (*AWS*) *Blockchain Partners Portal* provides a wide range of capabilities and the largest global infrastructure for building end-to-end blockchain platforms cost efficiently and on a large scale. *AWS Partner Network* (*APN*) Technology and Consulting partners offer a rapidly growing selection of blockchain and distributed-ledger solutions with support for multiple protocols.

The key features that AWS infrastructure offers to its users are:

- Easy-to-deploy development environment, already familiar to the community working with Cloud infrastructure. Facilitates the design of distributed ledgers and blockchain-based applications, reducing the time to plan and implement a basic setup, in particular when a large number of partners are involved.

- Scalability and vast amount of resources (pay as you grow) that can help a lot when testing out new limits in sharing information between nodes.
- The open source code and support available through the AWS blockchain partners. The reuse of code can speed up the basic proof-of-concept building and enables more diversity for testing different customer use-cases.

Several key players have developed their own distributed ledger solutions using the AWS platform mainly as IaaS. Most notable examples are Kaleido [65], Corda [52], and PokitDok [66].

5.1.5. Trusted IoT Alliance

Cisco, Bosch, and a number of other major IoT vendors launched the *Trusted IoT Alliance* in September 2017 to catalyze the development of blockchain-enabled, trusted IoT. The mission of the alliance is to bring companies together to develop and set the standard for an open source, interoperable blockchain protocol to support different IoT ecosystems on a global scale.

Trust for data produced by such IoT systems is provided in a blockchain-agnostic fashion, thereby enabling a decentralized trust model for interoperable digitized identities of physical goods, documents, immobilized assets, sensors, and machines. The ambition is to achieve performance and resilience that can scale to support billions of connected devices.

The alliance invites developers and enterprises to join and create proofs of concept and testbeds jointly with the partners onboard.

5.2. Data Delivery and Storage

5.2.1. Streamr

Streamr is a platform that utilizes blockchain technology to allow the creation of real-time data decentralized applications (Dapps) for IoT. Compared to IOTA, which was designed specifically for IoT, it leverages existing blockchain technology and builds upon that. It is, in essence, a decentralized peer-to-peer (P2P) network that claims to offer scalability, low-latency, untamperable data delivery, and persistence. The solution that the Streamr platform offers is *decentralized messaging* and *event processing*, while it aims to replace platforms such as Azure EventHub [67] and Azure StreamAnalytics [68].

The Streamr platform stack consists of the following five layers:

- **Streamr Editor:** A graphical user interface offering non-blockchain technology experts a toolbox for the development of Dapps.
- **Streamr Engine:** An event processing and analytics decentralized engine.
- **Streamr Data Market:** A universe of data streams that can be published or subscribed to. It allows the exchange and monetization of data. The data that traverse throughout the network are events (timestamped data) batched into streams; payments are made in *DATAcoin*, Streamr's internal cryptocurrency.
- **Streamr Network:** The data transport backbone and core of the platform. A P2P network whose basic software building block is the Broker Node.
- **Streamr Smart Contracts:** This is how the platform utilizes the blockchain technology. Smart contracts are used for incentivization, network coordination, permissioning, and integrity checking.

The Streamr Network layer combines the characteristics of systems such as Kafka [69], ZeroMQ [70], etc., which are scalable Cloud-based real-time data transport systems and decentralized P2P systems such as Whisper, Bitmessage, etc. In that way, it can offer high throughput for real-time data applications that the former type of systems achieve, while at the same time being able to effectively route data, discover peers, etc. that the latter systems support.

Through the Streamer Smart Contracts layer, the Network uses an underlying Ethereum stack (not limited to this specific technology), which is not used for data storage but rather for the following functions:

- **Stream registry:** Storing administrative information about data streams.
- **Network partitioning and coordination:** Reaching consensus on splitting the network into partitions and assigning them to specific broker nodes.
- **Incentivization:** Using DATAcoin as the network's usage token to subscribe to data streams and to reward broker nodes for carrying out their tasks; data integrity service via checksums and data delivery service.
- **Event persistence:** Storing the series of events offering robustness and fault tolerance.
- **Data provenance and data licensing:** Cryptographically sign data with a private key and grant access only to authorized users for a specific time period.

5.2.2. Flowchain

Flowchain [71] is a proposal for supporting peer-to-peer IoT networks and real-time transactions over a blockchain system. In Flowchain, each node can mine blocks, referred to as *virtual blocks*, in its own branch. Some of these blocks are valid, while others are invalid. One approach is to characterize the most recently used block of a branch as the only valid block of the specific branch. Only valid blocks contain IoT transaction data. Valid blocks can be merged using a branch merge algorithm to form a single blockchain. Such an approach for mining is used to support real-time data transactions, by avoiding the mining delay that exists in PoW blockchain systems, such as Bitcoin. Characterization of valid blocks considers a PoS mechanism. Chunks of IoT data, along with their hashes, are exchanged over the peer-to-peer network and are stored in the distributed data store between IoT devices. The forwarding of data chunks is based on the Chord [72] protocol.

Flowchain uses W3C's *Web of Things* (*WoT*) ontology as a standards-based model to represent a physical device as an application server, which can run on a high-performance device or a microcontroller. Moreover, a JavaScript runtime environment is used for high-performance devices, while a more lightweight environment is used for resource-constrained devices. Finally, Flowchain supports a gateway mechanism for interoperability between different ledgers within the same IoT peer-to-peer network.

5.2.3. Cyber-Physical Chain

Cyber-Physical Chain (*CPChain*) is a data platform for IoT systems supporting data acquisition, storage, sharing, and applications [73]. It separates the data, contract, and application layers from control, which supervises data interaction through a blockchain. Raw data are encrypted on the user side and stored in a distributed hash table (DHT). Only hash values, as unique identification of data, and credentials for integrity and correctness are published on the blockchain. Parallelization is supported by separating data storage from control. Re-encryption and homomorphic encryption technologies are combined to support one-to-many authorization.

CPChain plans to develop a hybrid consensus model that combines dynamic committee election with PoW consensus. The election process is performed in rounds. A node is eligible for election only if its credibility is above a threshold. However, it is not specified how a node's credibility is determined and updated.

To address the different requirements of IoT applications in terms of delay and security, sidechains implementing lightweight consensus protocols are proposed. For industrial scenarios, altruistic cooperative models are proposed.

5.3. Marketplace-Oriented Systems

5.3.1. IOTA Marketplace

IOTA Marketplace [74] is a pilot application of the IOTA technology. IOTA marketplace is in essence a data marketplace where users can purchase access to a sensor data stream. It currently supports tens of sensor types, mainly provided by the IOTA consortium members. Payments are made using the IOTA currency.

Sensor measurements are propagated in real time over the IOTA network using the *Masked Authenticated Messaging* (*MAM*). MAM is a data communication protocol which adds functionality to emit and access an encrypted data stream, like RSS, over Tangle [75] (IOTA's distributed ledger), regardless of the size or cost of the device. IOTA's consensus protocol adds integrity to these message streams [76].

Datum

Datum [77] is a distributed marketplace for data monetization that leverages an Ethereum-based smart contract platform. The platform allows anyone to securely and anonymously store and trade structured data collected from social networks, wearables, smart homes, and other IoT devices.

The Datum network claims to consist of 3 billion DAT tokens, with 1.53 billion available in a public token crowdsale. Up to 40% of raised funds is hedged in USD, EUR, and BTC. The Datum network aims to disrupt current data broker models by eliminating disintermediation in trading of social and IoT data.

Datum encompasses the following key elements:

- A decentralized data store allowing users to store structured data securely.
- The DAT token enabling payments for data storage and sharing.
- A data marketplace, enabling individuals to monetize their data on their terms.
- A mobile client application, available on Android and iOS.
- Datum leverages Ethereum, BigchainDB, and IPFS to provide a scalable, decentralized data storage backend.
- Data storage and data sharing is paid for by the DAT token. Data can be purchased as one-off or on an ongoing subscription basis. Interestingly, trading of DAT Tokens is prohibited for U.S. Citizens and residents in China and South Korea.

5.4. Timestamping Services

5.4.1. KSI Blockchain

KSI Blockchain is a globally distributed infrastructure for issuing and verifying *KSI signatures*. These signatures serve as a solid proof of *when* and *by whom* some data item of arbitrary size and format was generated. A user interacts with the KSI system by submitting the hash-value of the data to be signed. Then, the KSI system returns to the user a signature providing cryptographic proof of the *time of signature*, the *integrity of the signed data*, and the *data origin*, i.e., which entity generated the signature.

Unlike traditional digital signature approaches, such as the Public Key Infrastructure (PKI), which depend on asymmetric key cryptography, KSI uses only hash-function cryptography. Thus, the generation and verification of KSI signatures is based solely on the security of hash-functions and the availability of a public ledger, commonly referred to as a blockchain. KSI core technology is based on Buldas' and Saarepera's work [78].

The main benefits of the KSI system can be summarized as follows:

- **Unlimited Signing Rate:** KSI signatures can be generated at a practically unlimited rate. Even if an exabyte (i.e., 10^{18} bytes) of data are generated around the planet per second, and data records have an average size of 1MB, KSI is able to provide signatures for all one trillion (10^{12}) data records per second with negligible computational, storage, and network overhead.

- **Portability:** A KSI signature can be verified even after its respective data has crossed geographic or organizational boundaries and service providers.
- **Data Privacy:** KSI does not ingest any user data; data never leaves the user's premises. Instead, the system is based on one-way cryptographic hash functions that result in hash values uniquely representing the data while being irreversible. That is, one cannot start with the hash value and reconstruct the data, or come up with some alternative data that has the same hash value. Data privacy is guaranteed at all times.
- **Quantum Immunity:** The cryptography behind KSI signatures ensures that they will never expire and that they remain quantum-immune, i.e., secure even after the realization of quantum computing.
- **Independent Verification:** The properties of the signed data can be verified locally, without relying on a trusted authority.

5.5. Blockchain-Based IoT Applications

A growing number of IoT applications that leverage the power of blockchain technologies are already in use. *Tilepay* [79] offers a blockchain-based marketplace, where users can sell and buy data generated by Things they own. *Catenis* [80] provides Bitcoin-based secure messaging for the IoT. *Riddle&Code* [81], as well as *Chronicled* [82], provide a service that allows users to register real-world object identities in a blockchain (e.g., for claiming ownership or for tracking fraud). *Slock.it* [83] implements a blockchain-based marketplace where users can pay for services provided by real-world Things (e.g., a user can pay in virtual coins in order to open the door of a rented apartment). *Transactivegrid* [84], *Gridgularity* [85], and *SolarCoin* [86] combine smart grids with blockchains and they allow devices to record energy consumption, as well as to buy energy, all with the help of a blockchain. *Farmshare* [87] leverages blockchain to facilitate food donations by local farmers. *Provenance* [88] uses a blockchain to record events related to product supply chains, focusing on agricultural products.

6. Conclusions

The Internet of Things (IoT) and Distributed Ledger Technology (DLT) are two distinct new technologies that have evolved along separate paths motivated by different challenges. IoT was born by the need to build streamlined management frameworks for myriads of diverse sensors and devices that gained networking capabilities in the last couple of decades. DLT was motivated by the desire to provide provable trust in online transactions without depending on any centralized service or powerful institution. Interestingly, the two technologies have proven to blend strikingly well with each other, enabling the development of IoT ecosystems where smart devices of multiple parties, of possibly conflicting interests, collaborate seamlessly enjoying complete trust and accountability with respect to their interaction.

A number of distributed ledger designs have been proposed and deployed, targeting different application domains. We have identified four main characteristics that distinguish distributed ledgers into broad categories. First, the ledgers' access model, determining who can submit transactions, who can propose blocks, and who can read the blockchain. Second, the token model, that is, whether the blockchain encompasses a token for implementing a cryptocurrency and for allowing value representation and transfer. Third, the consensus protocol employed to let nodes validate transactions and reach global agreement on building and accepting blocks. Finally, the smart-contract functionality offered by a blockchain, ranging from simple scripting to more powerful smart contract programming languages that support Turing-complete computation.

Reviewing IoT technologies, one realizes immediately that there is a large number of IoT platforms, many of them being proprietary and either fully or partially closed. There have been various attempts for defining or describing in their generality IoT architectures and systems using a layered and modular approach for facilitating interoperability. There are also proposals that aim to achieve interoperability at a higher, semantic layer, compared to technical (connectivity-oriented) and syntactic interoperability, in order to enable different entities to exchange information, data, and knowledge in a meaningful way. Some IoT systems have already adopted blockchain technology.

These systems include industrial solutions, but also blockchain-community originating proposals that specifically target IoT data handling. More specifically, we have identified four general categories of frameworks merging IoT with blockchains: first, all-in-one systems, originating from organizations established in the IoT world; second, systems focusing on fast yet trusted and reliable data delivery and storage based on blockchains; third, systems oriented towards marketplace creation, enabling automated trading among IoT devices for various types of resources; and, finally, timestamping services, allowing an entity to timestamp their data, creating an indisputable proof of data ownership at a given point in time.

Acquiring an elaborate picture of the IoT and blockchain interplay landscape is a non-trivial task, given the plethora of solutions proposed and the diverse target application domains. This work aimed at unfolding a number of representative blockchain designs and blockchain-based IoT systems, serving as a starting point for exploring this promising blend of novel technologies.

Author Contributions: This paper is the result of collaborative work of all authors. S.V., N.F., V.A.S., and G.C.P. contributed to the entire paper, focusing on blockchain elements, blockchain implementations, oracles, and some of the IoT systems. M.J. and Y.O. contributed mainly on detailing certain IoT systems and on the oracles section.

Funding: This research has been undertaken in the context of project SOFIE (Secure Open Federation for Internet Everywhere), which has received funding from EU's Horizon 2020 programme, under grant agreement No. 779984.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Augusto, J.C.; Callaghan, V.; Cook, D.; Kameas, A.; Satoh, I. Intelligent Environments: a manifesto. *Human-Centric Comput. Inf. Sci.* **2013**, *3*, 12, doi:10.1186/2192-1962-3-12. [CrossRef]
2. Roalter, L.; Kranz, M.; Möller, A. A Middleware for Intelligent Environments and the Internet of Things. *Ubiquitous Intelligence and Computing*; Yu, Z., Liscano, R., Chen, G., Zhang, D., Zhou, X., Eds.; Springer: Berlin, Germany, 2010; pp. 267–281.
3. Ryu, M.; Kim, J.; Yun, J. Integrated semantics service platform for the Internet of Things: A case study of a Smart Office. *Sensors* **2015**, *15*, 2137–2160. [CrossRef] [PubMed]
4. Kim, H.M.; Laskowski, M. Toward an ontology-driven blockchain design for supply-chain provenance. *Intell. Syst. Account. Finance Manag.* **2018**, *25*, 18–27. [CrossRef]
5. Wollschlaeger, M.; Sauter, T.; Jasperneite, J. The future of industrial communication: Automation networks in the era of the Internet of Things and Industry 4.0. *IEEE Ind. Electron. Mag.* **2017**, *11*, 17–27. [CrossRef]
6. Stojkoska, B.R.; Trivodaliev, K. A review of Internet of Things for smart home: Challenges and solutions. *J. Clean. Prod.* **2017**, *140*, 1454–1464. [CrossRef]
7. Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. Blockchain for IoT security and privacy: The case study of a smart home. In Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kona, HI, USA, 13–17 March 2017; pp. 618–623.
8. Lv, Z.; Halawani, A.; Feng, S.; Ur Réhman, S.; Li, H. Touch-less Interactive Augmented Reality Game on Vvision-based Wearable Device. *Personal Ubiquitous Comput.* **2015**, *19*, 551–567. [CrossRef]
9. Catarinucci, L.; De Donno, D.; Mainetti, L.; Palano, L.; Patrono, L.; Stefanizzi, M.L.; Tarricone, L. An IoT-aware architecture for smart healthcare systems. *IEEE Internet Things J.* **2015**, *2*, 515–526. [CrossRef]
10. Mohanty, S.P.; Choppali, U.; Kougianos, E. Everything you wanted to know about Smart Cities: The Internet of Things is the Backbone. *IEEE Consum. Electron. Mag.* **2016**, *5*, 60–70. [CrossRef]
11. Li, S.; Xu, L.; Zhao, S. The Internet of Things: A Survey. *Inf. Syst. Front.* **2015**, *17*, 243–259. [CrossRef]
12. Whitmore, A.; Agarwal, A.; Xu, L. The Internet of Things—A survey of topics and trends. *Inf. Syst. Front.* **2015**, *17*, 261–274. [CrossRef]

13. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
14. Christidis, K.; Devetsikiotis, M. Blockchains and Smart Contracts for the Internet of Things. *IEEE Access* **2016**, *4*, 2292–2303. [CrossRef]
15. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An overview of blockchain technology: Architecture, consensus, and future trends. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017; pp. 557–564.
16. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: http://bitcoin.org/bitcoin.pdf (accessed on 20 September 2019).
17. Ethereum. Available online: https://ethereum.org (accessed on 20 September 2019).
18. Litecoin. Available online: https://litecoin.org/ (accessed on 20 September 2019).
19. Reed, J. *Litecoin: An Introduction to Litecoin Cryptocurrency and Litecoin Mining*; CreateSpace Independent Publishing Platform: Scotts Valley, CA, USA, 2017.
20. Monero. Available online: https://www.getmonero.org/ (accessed on 20 September 2019).
21. Dash. Available online: https://www.dash.org/ (accessed on 20 September 2019).
22. Dogecoin. Available online: https://www.dogecoin.com/ (accessed on 20 September 2019).
23. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; doi:10.1145/3190508.3190538. [CrossRef]
24. Deetman, S. Bitcoin Could Consume as Much Electricity as Denmark by 2020. Available online: https://motherboard.vice.com/en_us/article/aek3za/bitcoin-could-consume-as-much-electricity-as-denmark-by-2020 (accessed on 20 September 2019).
25. QuantumMechanic. Proof of Stake. Available online: https://bitcointalk.org/index.php?topic=27787.0 (accessed on 20 September 2019).
26. Cardano. Available online: https://whycardano.com (accessed on 20 September 2019).
27. The Lisk Protocol. Available online: https://lisk.io/documentation/lisk-protocol (accessed on 20 September 2019).
28. Blackcoin. Available online: https://blackcoin.org/blackcoin-pos-protocol-v2-whitepaper.pdf (accessed on 20 September 2019).
29. King, S.; Nadal, S. Peercoin—Secure & Sustainable Cryptocoin. Available online: https://peercoin.net/whitepaper (accessed on 20 September 2019).
30. Nxt White Paper. Available online: https://bravenewcoin.com/assets/Whitepapers/NxtWhitepaper-v122-rev4.pdf (accessed on 20 September 2019).
31. Proof of Authority. Available online: https://github.com/paritytech/parity/wiki/Proof-of-Authority-Chains (accessed on 20 September 2019).
32. Angelis, S.D.; Aniello, L.; Baldoni, R.; Lombardi, F.; Margheri, A.; Sassone, V. PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain. In Proceedings of Italian Conference on Cyber Security, Milan, Italy, 6–9 February 2018.
33. Pease, M.; Shostak, R.; Lamport, L. Reaching Agreement in the Presence of Faults. *J. ACM* **1980**, *27*, 228–234. [CrossRef]
34. Lamport, L.; Shostak, R.; Pease, M. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* **1982**, *4*, 382–401. [CrossRef]
35. Castro, M.; Liskov, B. Practical Byzantine fault tolerance. *OSDI* **1999**, *99*, 173–186.
36. Intel: Sawtooth Lake. Available online: https://intelledger.github.io/ (accessed on 20 September 2019).
37. Hyperledger Sawtooth. Available online: https://www.hyperledger.org/projects/sawtooth (accessed on 20 September 2019).
38. Siris, V.A.; Dimopoulos, D.; Voulgaris, S.; Fotiou, N.; Polyzos, G.C. OAuth 2.0 Meets Blockchain for Authorization in Constrained IoT Environments. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 15–18 April 2019.
39. Buterin, V. A Next-Generation Smart Contract and Decentralized Application Platform. Available online: https://github.com/ethereum/wiki/wiki/White-Paper (accessed on 20 September 2019).

40. Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Available online: https://github.com/ethereum/yellowpaper (accessed on 20 September 2019).
41. Kiayias, A.; Miller, A.; Zindros, D. Non-Interactive Proofs of Proof-of-Work. Available online: https://eprint.iacr.org/2017/963 (accessed on 20 September 2019).
42. Low-Level Virtual Machine (LLVM): a compiler technologies umbrella project. Available online: https://llvm.org (accessed on 20 September 2019).
43. IOTA. Cryptographic Vulnerabilities in IOTA. Available online: https://medium.com/@neha/cryptographic-vulnerabilities-in-iota-9a6a9ddc4367 (accessed on 20 September 2019).
44. Heilman, E.; Narula, N.; Dryja, T.; Virza, M. IOTA Vulnerability Report: Cryptanalysis of the Curl Hash Function Enabling Practical Signature Forgery Attacks on the IOTA Cryptocurrency. Available online: https://github.com/mit-dci/tangled-curl/blob/master/vuln-iota.md (accessed on 20 September 2019).
45. Hyperledger Indy. Available online: https://www.hyperledger.org/projects/hyperledger-indy (accessed on 20 September 2019).
46. Decentralized Identifiers (DIDs) v0.12 Data Model and Syntaxes for Decentralized Identifiers (DIDs). Available online: https://w3c-ccg.github.io/did-spec/ (accessed on 20 September 2019).
47. Sovrin. Available online: https://sovrin.org/ (accessed on 20 September 2019).
48. Aublin, P.; Mokhtar, S.B.; Quéma, V. RBFT: Redundant Byzantine Fault Tolerance. In Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems, Philadelphia, PA, USA, 8–11 July 2013; pp. 297–306.
49. MultiChain. Available online: https://www.multichain.com/ (accessed on 20 September 2019).
50. CoinSpark. Available online: http://coinspark.org (accessed on 20 September 2019).
51. Ali, M.; Nelson, J.C.; Shea, R.; Freedman, M.J. Blockstack: A Global Naming and Storage System Secured by Blockchains. In Proceedings of the USENIX Annual Technical Conference, Denver, CO, USA, 22–24 June 2016; pp. 181–194.
52. Corda. Available online: https://www.corda.net/ (accessed on 20 September 2019).
53. Hearn, M. Corda: A distributed ledger. Available online: https://www.corda.net/content/corda-technical-whitepaper.pdf (accessed on 20 September 2019).
54. Blockchain Oracles Will Make Smart Contracts Fly. Available online: https://hackernoon.com/oracles-help-smart-contracts-resolve-subjective-events-d81639d8291c (accessed on 20 September 2019).
55. Zhang, F.; Cecchetti, E.; Croman, K.; Juels, A.; Shi, E. Town Crier: An Authenticated Data Feed for Smart Contracts. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 270–282. doi:10.1145/2976749.2978326. [CrossRef]
56. Oraclize. Available online: https://oraclize.it (accessed on 20 September 2019).
57. Benet, J. IPFS—Content Addressed, Versioned, P2P File System. *arXiv* **2014**, arXiv:1407.3561.
58. TLSNotary—A Mechanism for Independently Audited HTTPS Sessions. Available online: https://tlsnotary.org/TLSNotary.pdf (accessed on 20 September 2019).
59. ChainLink. Available online: https://www.smartcontract.com/link (accessed on 20 September 2019).
60. Hivemind. Available online: http://bitcoinhivemind.com (accessed on 20 September 2019).
61. Off-Chain Computation Solutions for Ethereum Developers. Available online: https://medium.com/@YondonFu/off-chain-computation-solutions-for-ethereum-developers-507b23355b17 (accessed on 20 September 2019).
62. TrueBit. Available online: https://truebit.io (accessed on 20 September 2019).
63. Corporation, M. The Coco Framework Technical Overview. Available online: https://github.com/Azure/coco-framework/blob/master/docs/Coco%20Framework%20whitepaper.pdf (accessed on 20 September 2019).
64. Corporation, H. Hdac: Transaction Innovation—IoT Contract & M2M Transaction Platform Based on Blockchain, Whitepaper. Available online: https://github.com/Hdactech/doc/wiki/Whitepaper (accessed on 20 September 2019).
65. Kaleido. Available online: https://kaleido.io/ (accessed on 20 September 2019).
66. PokitDok. Available online: https://pokitdok.com/ (accessed on 20 September 2019).
67. Azure Eventhub. Available online: https://docs.microsoft.com/en-us/azure/event-hubs/ (accessed on 20 September 2019).

68. Azure StreamAnalytics. Available online: https://azure.microsoft.com/en-us/services/stream-analytics/ (accessed on 20 September 2019).
69. Kreps, J.; Corp, L.; Narkhede, N.; Rao, J.; Corp, L. Kafka: A Distributed Messaging System for Log Processing. In Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB), Athens, Greece, 12–16 June 2011.
70. Hintjens, P. ZeroMQ: An Open-Source Universal Messaging Library. Available online: http://zguide.zeromq.org/page:all (accessed on 20 September 2019).
71. Chen, J. Flowchain: A Distributed Ledger Designed for Peer-to-Peer IoT Networks and Real-Time Data Transactions. Available online: https://www.flowchain.co/Flowchain-WhitePaper.pdf (accessed on 20 September 2019).
72. Stoica, I.; Morris, R.; Karger, D.; Kaashoek, M.F.; Balakrishnan, H. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, San Diego, CA, USA, 27–31 August 2001; pp. 149–160.
73. Long, C.; Zhao, B.; Shi, Q. Decentralized Infrastructure for Next Generation Internet of Things. Available online: https://www.cpchain.io/CPChain_Whitepaper_English.pdf (accessed on 20 September 2019).
74. IOTA Marketplace. Available online: https://data.iota.org/ (accessed on 20 September 2019).
75. Popov, S. The Tangle, White Paper. Available online: https://iotatoken.com/IOTA_Whitepaper.pdf (accessed on 20 September 2019).
76. Masked Authenticated Messaging. Available online: https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e (accessed on 20 September 2019).
77. Haenni, R. Datum Network: The Decentralized Data Marketplace. Available online: https://blockstack.org/whitepaper.pdf (accessed on 20 September 2019).
78. Buldas, A.; Truu, A.; Laanoja, R.; Gerhards, R. Efficient Record-Level Keyless Signatures for Audit Logs. In *Nordic Conference on Secure IT Systems*; Springer: Berlin, Germany, 2014; pp. 149–164.
79. Tilepay. Available online: http://www.tilepay.org (accessed on 20 September 2019).
80. Catenis. Available online: http://blockchainofthings.com (accessed on 20 September 2019).
81. The Blockchain Interface Company. Available online: https://www.riddleandcode.com/ (accessed on 20 September 2019).
82. Chronicled. Available online: http://www.chronicled.com/ (accessed on 20 September 2019).
83. Slock.it. Available online: https://slock.it (accessed on 20 September 2019).
84. LO3 Energy. Available online: https://lo3energy.com/ (accessed on 20 September 2019).
85. Gridgularity. Available online: http://gridsingularity.com (accessed on 20 September 2019).
86. SolarCoin. Available online: https://solarcoin.org (accessed on 20 September 2019).
87. Farmshare. Available online: http://farmshare.org (accessed on 20 September 2019).
88. Provenance. Available online: https://www.provenance.org/ (accessed on 20 September 2019).