cryptography

Article

Beyond Bitcoin: A Critical Look at Blockchain-Based Systems

Diego Romano and Giovanni Schmid *

Istituto di Calcolo e Reti ad Alte Prestazioni, I 80131 Naples, Italy; diego.romano@cnr.it
* Correspondence: giovanni.schmid@cnr.it; Tel.:+39-08-1613-9529

Received: 7 July 2017; Accepted: 16 August 2017; Published: 1 September 2017

Abstract: After more than eight years since the launch of Bitcoin, the decentralized transaction ledger functionality implemented through the blockchain technology is being used not only for cryptocurrencies, but to register, confirm and transfer any kind of contract and property. In this work, we analyze the most relevant functionalities and known issues of this technology, with the intent of pointing out the possible behaviours that are not as efficient and reliable as they should be when thinking with a broader outlook.

Keywords: cryptocurrency; blockchain technology; smart contract

1. Introduction

The existence of digital currencies is strictly related to the broad diffusion of Internet and on-line markets. A digital currency is similar to electronic money but even if it can be used to buy services or physical goods like traditional money, it is not equivalent or linked to any fiat currency. The idea of using a digital currency became realistic when some important developments in cryptography settled multiple fundamental security challenges in money transfers, e.g., the necessary trust in the transaction. Several digital currencies have been implemented in the last 20 years, but only starting in 2008 did the project for a currency decentralized in every respect become a reality. In fact previous implementations (e.g., [1–4]) employed a point of control over the money supply, which in some cases exposed the whole system to attacks for taking control over it. This happened for both governmental and criminal interests [5,6].

Bitcoin [7] is the first implementation of a *cryptocurrency*, a concept first described in 1998 [8] on the *cypherpunks* mailing list, a distributed and anonymous electronic mailing list used intensively during the nineties by activists advocating the widespread use of strong cryptography and privacy-enhancing technologies as a route to social and political change. By digitally emulating cash creation and transactions through cryptographic operations, Bitcoin gets rid of bank charges and tries to anonimize the involved parties. It includes a peer-to-peer network, with both a distributed currency issuance system and a transaction verification system, backed by a public tansaction ledger called *blockchain*.

Since January 2009, when Bitcoin v0.1 was released and announced on a cryptography mailing list, an interest in its features has been growing within various communities related in some way to Information Technology. Nowadays, Bitcoin has a Market Cap of about 42,149,807,705 USD [9], and a plethora of alternate cryptocurrencies (*altcoins*) have been launched on the market. Although many of these projects were unsuccessful business ventures and are now defunct, some others seem grounded on durable entrepreneurial ideas and/or smart innovation (for a nice visual history of this buzzing world until 2015, see [10]). Generally speaking, it has become evident that the decentralized transaction ledger functionality implemented through the blockchain technology can be used not only for cryptocurrencies, but to register, confirm and transfer any kind of contract and property. Clearly, such approach can be profitably adopted in many application scenarios and industry sectors, from logistic to finance, in healthcare, or even as a communication framework for supporting Artificial Intelligence (AI) applications. For example, public records such as vehicle registrations or marriage certificates could be migrated to suitable blockchains. Blockchain technology enables indeed the creation and management of *smart contracts* and *smart properties* [11]. On the other hand, these two new digital paradigms can be used to give rise to the creation of *decentralized autonomous organizations* [12].

In this work we will consider the opportunities given by the blockchain, analysing its most relevant functionalities and known issues. We will evaluate the computational features specific to Bitcoin, with the intent of showing the unbearableness of Bitcoin-derived systems on a global scale.

More in details: in Section 2, we will review some new potential applications, surveying the actual tools that are already implemented and focusing on their functionalities related to the blockchain. Then in Section 3 we will stress the role of the transaction concept in the actualization of the blockchain, followed by an inspection of the components used to manage the public ledger, with a special look at the case of Bitcoin. A discussion on some computational aspects will be presented in Section 4, where some serious inefficiencies will be highlighted. Finally, in Section 5 we will draw our conclusions.

2. Current and Potential Applications

Considered as a public ledger, in its nature the blockchain is inclined to record formal agreements between parties. This is the case when a contract must be registered by a notary, or when the ownership of something must be publicly advertised and recognized, or even when a property with some automatic behaviour could interact with another subject enforcing a contract. These are loose examples—different from the original recordings of currency transactions—that show some potential applications of the blockchain, assuming that we properly replace the attributes of its records. This approach does not take into account the inner functionalities of the underlying tools, but rather assumes it is a secure and efficient system. In the following we introduce some interesting real applications in this perspective, leaving other more complex cases to next sections.

2.1. Smart Properties

The key idea of *smart property* [13] is the assertion of ownership rights for an asset through its registration in the blockchain, secured by means of a private key. Only someone who has got the private key can ascertain its ownership of the asset, which on the other hand can be verifed by anyone thanks to the corresponding public key. The owner could later sell the asset by giving its corresponding private key to another user.

Some types of property like trademarks, copyrights and patents are inherently smart properties, and their management as such is natural and can be fruitful. Indeed these entitlements can be easily encoded and processed as digital documents. This is not the case with physical assets, where ownership is more exposed to frauds. In fact in order to register a chattel (e.g., a car) in the blockchain, we need to attach a uniquely identifiable tag or a chip to it. If the information contained in that tag or chip could be altered, or the tag/chip could be detached, smart property would not be guaranteed.

An actual implementation of smart properties is *colored coins*. Using this term we are talking about a class of methods for representing and managing real world assets on top of the Bitcoin blockchain. In fact it is possible to store a small amounts of metadata on the blockchain to represent asset manipulation instructions. One can encode in a Bitcoin transaction the information that some units of a new asset were issued and credited to a given Bitcoin address, and a real world value can correspond to those units with the issuer's promise to redeem them for some goods or services. For example, a car dealer can issue 100 units of a "Buy car X during the time promotional period Y" asset, and promise to redeem each unit for a right to buy a certain kind of car as part of a promotional offer [14].

Blockchain ability to store information such as identity, ownership, membership, etc., enable governments to provide services for citizens related to birth and death certificates, business licenses and property titles [15]. Information is stored on a blockchain and can be accessed based on predefined terms. Entities are able to access the blockchain and provide certain information to the decentralized system by generating public-private key pairs. By holding with confidentiality protection its private keys, an entity is in control of its identity and the related information. The public keys work instead as addresses where information can be looked up. One real life example is the one created by BitNation (bitnation.co) which aims to initiate decentralized governance at global scale such as World Citizenship ID, and a Refugee Emergency Response project, featuring a BitNation Emergency ID (BE-ID) for displaced individuals in the ongoing refugee crises in Europe.

2.1.1. Proof of Existence

An interesting implementation of the smart property concept in the context of authorship protection is Proof of Existence [16]. Proof of Existence is a web-based service used to prove the authorship of things such as software or documents. This tool demonstrates document ownership without revealing the information it contains, and it can be used to prove that a document was created at a certain time. When a user requires the proof of existence for a given digital document, the service does not record the content of the document but instead computes a digest of its content thanks to a cryptographic hash function. Later, that digest is inserted into a block of the blockchain, and the block timestamp becomes the document timestamp. If a user tries to insert a modified version of the same document, a new digest is created which—with overhelming probability—will be completely unrelated to the previous one, because of the properties of cryptographic hash functions.

2.1.2. Everledger

Another example for smart property is Everledger [17], which was introduced as a global digital registry for diamonds. By using more than 40 features, including color and clarity, a diamond's digital ID can be created and used as a certificate to be recorded on the blockchain, therefore chronicling the jewel's ownership, from mine to ring. This idea became successful because it is used within the Kimberley Process Certification Scheme [18]. With the aim of preventing diamonds mined in war zones to be brought illicitly into the market in their uncut form, Everledger digital ID builds a reputation for every single piece. At the date of writing, more than a million diamonds were recorded, but other precious goods are taken into consideration. For examples, an ID for fine wines can be created by using unique characteristics such as the cork, the label and the bottle.

2.2. Smart Contracts

Smart contracts represent the implementation of a contractual agreement, whose legal provisions are formalized into programming code and verified through a network of peers [19]. Indeed these contracts are defined through the code and executed or enforced by the code, without the need for a trusted third party. An example of a smart contract is the enforcement of a bet between two users about the maximum humidity level tomorrow. On the following day the contract is automatically completed by a software program checking the humidity levels provided by a qualified weather service or some given sensors, as stated by the contract itself, reading and transferring funds from the loser's to the winner's account. Another example is an inheritance gift that becomes available on the child's eighteenth birthday. As should be clear from these examples, in order to set up a smart contract one needs to choose an event or condition which triggers the transaction expected in the contract, and then check with a program that the event or condition has occurred.

According to [11], smart contracts represent a technological advancement to the practice of law, which allow contracting parties to structure their relationships more efficiently, in a self-executing way and without the ambiguity of natural words. Given the reliance on source code, users can model contractual performance and simulate the agreement effectiveness before its execution [19].

We can point out the following three distinctive properties of smart contracts: autonomy, self-sufficiency and decentralization [20]. Smart contracts are *autonomous* in the sense that, after their deployment on the blockchain, they operate without any human intervention. Furthermore, they can accumulate capital over time, such as digital currencies or physical assets, and that is what the term *self-sufficient* stands for. Finally, smart contracts are *decentralized* because they are distributed and self-executing across a network of peer nodes.

Some open source projects have been started in the last years to develop programming languages for the easy creation of smart contracts (e.g., [21,22]). Using these programming languages, users will be able to get increasingly sophisticated smart contracts, designed to satisfy various requirements and to be used in several contexts.

In the following, two relevant projects related to smart contracts are discussed briefly in order to gain some insight on goals and functions for such blockchain-based systems.

2.2.1. Namecoin

Currently, the mapping between IP-addresses and the names of devices and services over the Internet implemented through the Domain Name Service (DNS) is supervised by the ICANN [23]. ICANN is a central authority whose main task is to establish the *top-level domain names* (TLD) (e.g., ".com", ".it", etc.), and which third party companies, known as *registrars*, are permitted to manage names in each TLD. In fact each registrar deals with accepting domain name orders and customer's requests for Internet services in the TLDs of its competence.

Government agencies could contact a registrar and induce it to blacklist some devices and services, for the purpose of political censorship. The extent of Internet censorship varies on a country-to-country basis: most democratic countries have moderate Internet censorship, whilst other countries go as far as to limit the access of information and suppress discussion among citizens [24].

Namecoin [25] is a blockchain-based DNS that—at least in the intentions of its designers and promoters—cannot be controlled by any government or organization. A blockchain-based DNS system means that DNS lookup tables are recorded on a blockchain and shared this way on a peer-to-peer system, so that naming in TLDs can be managed cooperatively by customers, without control by any registrar. Since domain name registration is nothing else than a special kind of contract between a registrar and its customers, Namecoin is a service for supplying smart contracts which is based on the Bitcoin cryptocurrency. Namecoin has the unique TLD `.bit`, and domains can be registered directly with the Namecoin system or via registration services like https://dotbit.me/. Since `.bit` is not managed by ICANN, in order to resolve names within Namecoin tables, one must set up a `.bit` proxy server for the correct handling of those DNS requests.

2.2.2. Ethereum

Ethereum aims at being a platform for the rapid deployment of generic decentralized blockchain-based systems, where etherogeneous applications can interact very efficiently. The Ethereum framework of concepts was proposed in late 2013 by Vitalik Buterin, a programmer involved in the Bitcoin project. Having failed to gain agreement on a Bitcoin scripting language for the development of custom applications, Buterin stepped forward by introducing a scripting language allowing anyone to write smart contracts and decentralized applications with arbitrary rules for ownership, transaction formats and state transition functions. Ethereum, by implementing a generic programmable blockchain, enables the execution of stateful, fully-customizable smart contracts. This way it can be used for deploying voting systems, domain name registries, financial exchanges, crowdfunding platforms, company governance, intellectual property and smart property (see Section 2.1).

Ethereum can be described as a virtual machine which makes use of a blockchain with a built-in programming language for developing and deploying distributed applications; what takes care of the internal state and the computations as specified in the blockchain is indeed the so-called *Ethereum Virtual Machine* (EVM). Basically, the EVM works in the same way as any other virtual machine: it takes some high level programming language designed for writing smart contracts, and compiles it into EVM bytecode that the computer, on which it runs, understands. The EVM can be thought of as a large decentralised computer containing millions of objects called *accounts*. Accounts have the ability to maintain an internal database, execute code and talk to other accounts. There are two types of accounts: *externally owned accounts* (EOAs) and contract accounts. *Contract accounts* are collections of code and data with specific addresses on the Ethereum blockchain. They are able to exchange messages between themselves and doing computations, and they serve to implement generic smart contracts. Instead, EOAs represent identities of external agents (humans, mining nodes or automated agents). They are the counterpart of Bitcoin's wallets, and use public key cryptography to sign transaction so that the EVM can securely validate the identity of the transactor. If you own the signing key(s) belonging to a given EOA, you have the ability to send *ether*—the Ethereum's cryptocurrency—and messages from such EOA to other accounts in the system [26].

Ethereum is first and foremost an ambitious and complex open-source software project, started at the beginning of 2014 and supported by various entrepreneurial initiatives over time. After a year and a half of development resulting in nine proof-of-concept prototypes and in a testbed network named Olympic, the Ethereum Foundation announced in July 2015 the Frontier network [27] as the first experimental live release of the Ethereum network. Since then, after further refinements and improvements, a stable release named Homestead was issued on March 2016, which is the current release at the time of writing. In June 2016, however, a major exploit targeted the Ethereum network: 3.6 million ethers (about 70 million USD at the time) were stolen by some accounts of *The DAO*, a form of investor-directed venture capital implemented as a complex smart contract with many features on Homestead, and transferred to an account of an anonymous entity. The exploit realized one of the attacks described in [28], a technical report appeared three weeks before on GoogleDocs. The authors therein analyze the rules of The DAO and identify nine causes for concern that can lead participants to engage in strategic rather than honest behaviors, with some behaviors that can cause honest investors to have their investments hijacked or committed to proposals against their interest and intent.

The DAO—which at the time was the most prominent project based on Ethereum, with a crowdfounded capital of about 150 million USD—collapsed because of the above circumstances. Moreover, the community decided to do a hard fork in the Ethereum blockchain, for the sole purpose of returning all the ethers taken from The DAO contract account to a new smart contract, through which they would be restored over time to the owners. However, the above decision created a lot of controversy among the Ethereum community, which on July 2016 split in two different networks, Ethereum (ETH) and Ethereum Classic (ETC) [29]. At the time of writing Ethereum Classic still offers the same features and it has all the same specifications (average block time, size, reward, etc.) as Ethereum, but the two networks will probably diverge in the next future. Statistics for the ETH blockchain are available at [30,31], whilst two major explorers for the ETC blockchain are [32,33].

The above drawbacks do not seem to have diminished interest and expectations of Finance and Industry, which see in Ethereum a backbone for the creation of various kinds of blockchain-oriented distributed systems and services. In March 2017, various blockchain start-ups, research groups, and Fortune 500 companies (including Cisco, Intel, J.P. Morgan and Microsoft) announced the constitution of the Enterprise Ethereum Alliance (EEA), with the vision of creating an open source standard rather than a product for addressing enterprise deployment requirements, which evolves in tandem with advances in public Ethereum and leverages existing standards [34].

2.3. Decentralized Autonomous Organizations

The blockchain technology makes the execution of several smart contracts and smart properties possible, enabling their reciprocal interaction in a decentralized and distributed way. *Decentralized autonomous organizations* (DAOs) [12] operate according to rules and procedures defined by smart contracts, and on the basis of ownerships defined through smart properties. In these organizations, machines and people can cooperate without the need to be incorporated into traditional business identities.

The decentralized autonomous organization concept derived from AI: a decentralized network of autonomous agents perform tasks, which can be conceived in the model of a corporation running with only a collateral human involvement under the control of a set of business rules. Such organizations can charge users for the services they provide, in order to pay others for the resources they need. As long as they receive sufficient funds to operate on their own, they can thus subsist independently of any third party.

If a decentralized organization is truly autonomous, no one (including its original creator) can control it after it has been deployed on the blockchain. That is actually a very threatening property: an organization could be conceived to evolve over time in order to adapt to the context in which it operates, thus it can be very difficult to know a priori if it can assume a dangerous state.

In the following we will present two examples of DAO: the first is Storj, a decentralized cloud storage platform which fits perfectly in the definition above; the second, ADePT, tends more to AI and the Internet of Things (IoT).

2.3.1. Storj

Storj [35] consists in a platform, supplied with a cryptocurrency and a suite of decentralized applications, to store data in a secure and decentralized manner. The files are encrypted on client-side, shredded into little pieces called "shards", and stored in a decentralized network of peers.

Indeed, the Storj protocol creates a distributed network for the formation and execution of storage contracts between peers: it includes contract negotiation, data transfer, remote data verification of integrity and availability, data retrieval, and payments. Each peer is an autonomous agent, capable of performing these actions without significant human interaction, which suggests the classification of Storj within the decentralized autonomous organizations.

Traditional cloud storage is vulnerable to a variety of security threats, including man-in-the-middle attacks, malware, and application flaws. Moreover, because many storage devices rely on the same infrastructure, failures are correlated across files and systems. Storj, instead, maintain data security using client-side encryption, while data integrity is implemented via a proof of retrievability. Therefore, data on the network should be resistant to censorship, tampering, unauthorized access, and also data failures since every shard has multiple copies on different storage nodes.

From the economic point of view, the basic idea of this DAO is to drive down the costs for various storage services by enabling more parties to compete using existing devices.

2.3.2. ADePT

From a different point of view, the concept of decentralized autonomous organization can actually be linked to the development of the IoT. The IoT will consist of billions of Internet-enabled devices, but not all can be blindly trustworthy and some could be even malicious. In order to facilitate private, secure, and (trustless) machine-to-machine coordination, these devices will need a central reference point. In this context the *Autonomous Decentralized Peer-to-Peer Telemetry* (ADePT) [36], by IBM and Samsung, represents a proof of concept system for the next generation of the IoT. It uses blockchain technology to provide the backbone of the system, utilizing a mix of peer-to-peer protocols to get secure and fault-tolerant transactions. In ADePT, the blockchain is seen as a way for devices to understand what other devices do, and the instructions and permissions different users have around these devices. In practice this can mean tracking relationships between a user and a device, and even between two devices, with the consent of the user.

In order to realize the ADePT concept, IBM and Samsung chose three protocols:

- **Ethereum**, in order to allow devices to understand contracts and capabilities (this is where blockchain technology comes into play);
- **Telehash** [37], a private messaging protocol used to share information among two or more devices;
- **Bit Torrent** [38] a file sharing protocol used to move data around also in case of discontinuous and unreliable connections.

According to [36], a Samsung W9000 washing machine was reconfigured to work within the ADePT system. When needed, the machine uses smart contracts to issue commands to a detergent retailer in order to receive new supplies. Thanks to these contracts the device has the ability to pay for the order itself, and later receive a notice from the retailer that the detergent has been paid for and shipped. Moreover, if a smartphone is connected to the home network, this information is broadcasted to it, presenting the purchase details to the owner of the washer.

2.4. Other Possible Application Fields

Going beyond the tools so far presented, a plethora of ideas for possible applications exists. Blockchain technology represents an opportunity to alleviate inefficiencies caused by third party trust organizations, logistics processing time, cumbersome, costly and risky correspondent networks. The adoption of such opportunity should help organizations to save costs and time as well as increase security for online transactions of any kind [39].

In the following we will outline some of the most interesting ideas in the fields of eScience, healthcare and financial services.

2.4.1. eScience

A context where the blockchain could be revolutionary in the future is Science. As we will show in detail in Section 4, instead of using an enormous consumption of electricity to crunch arbitrary numbers for mining, computations could be used to more practical tasks like solving existing science problems.

Another interesting aspect pertains the trust in scientific research. It is an important factor for the credibility of the scientific outcomes, especially in vital areas such as medical sciences. In the past, misconduct related to data manipulations such as outcome switching, data cleansing and selective results publication caused lack of confidence in scientific results.

A study by Carlisle in 2014 [40] has showed that blockchain can offer a low cost, independently verifiable method to audit and confirm the reliability of the results of scientific studies by using Bitcoin blockchain. Indeed, blockchain provides an immutable record of the existence, integrity and ownership of a specific medical trial protocol.

2.4.2. Healthcare

Besides the aspects related to medical trials, blockchain-based systems offer benefits also for the healthcare industry. By using their ability to provide trust and security, new models for managing and sharing medical records have emerged, mainly because of costs, time and resources required by traditional health management infrastructure [41]. For example, the government of Estonia established a partnership with a cyber-security firm called Guardtime (https://guardtime.com) to make use of their KSI ® technology based on blockchain in order to authenticate and verify the integrity of medical public data.

The diffusion of wearables in the health care is still not pervasive, also because of concerns about data security. For this reason, some initiatives like Healthbank (https://www.healthbank.coop) and Netcetera (https://www.netcetera.com) in Switzerland as well as Noser (https://www.noser.com) in Germany are investing on blockchain-based systems to securely share personal medical data.

Also the access to genetic data can be managed through blockchain, with the aim of discovering a predisposition for some diseases such as Alzheimer. For example, a blockchain could be used in genomics because it provides a structure for storing health data which should be later analyzed remaining private. This may be very useful in countries where people do not have the right to access their own genetic data. Blockchain-based genomic services could provide low cost genomic sequencing to individuals, making data available using their private key [20].

2.4.3. Financial Services

One of the sectors showing increasing interest in blockchain-based systems is the one of financial services. Thanks to the perception of the blockchain as an alternative to the current approach for conducting transactions, the blockchain idea seduced many institutions and banks like JP Morgan and Goldman Sachs. Through specific partnerships initiated to develop systems according to their needs, standards and expectations, financial institutions aim at eliminating centralized trust agencies. Bank Santander claims that distributed ledgers could save the banking industry $20 billion a year by 2022 [42]. Moreover, blockchain could result useful in several aspects [43,44]:

- In the scope of loans, usually banks along with credit scoring and rating firms assist in the issuance of mortgages, bonds, securities, etc. On the blockchain, anyone could check creditworthiness before issuing, trading and settling traditional debt instruments directly from peers.
- Blockchain cuts the settlement period on transactions from days to hours or minutes. The adoption of such tool in trading could reduce the post-trade processing such as settlement and reconciliation, which would eventually result in cost reduction.
- Each financial institution maintains its own ledgers for its assets (often product and/or region specific), which implies a multiplication of registers among parts. The reconciliation process of these ledgers is costly especially in the case of large banks, and software solutions often rely upon coding languages as awkward and primitive as Visual Basic for Applications (VBA). This results in human error, inefficiencies and therefore a decreased ability to manage risk. Creating databases among financial institutions resting upon a blockchain, could reduce these frictions.
- Smart contracts have the potential to automate existing logic in contracts which have multiple payment strands. As the majority of financial assets exist only in electronic form, many products could be governed by smart contracts.
- Compliance with regulation is costly, and a regulator enabled to view a transparent ledger shared by the financial services industry could drastically reduce the cost of Anti Money Laundering and fighting against terrorism financing.

Nowadays, in order to securely exchange information about their financial transactions, financial institutions use a global system maintained by SWIFT (Society for Worldwide Interbank Financial Telecommunication). Swift is globally used by more than 11,000 financial institutions in 209 countries to exchange $5 trillion a day, and therefore it is difficult for banks and other financial organizations to make a transition from Swift to a blockchain-based system. Meanwhile, Ripple [45] is a blockchain-savvy company which offers a so-called Interledger Protocol for connecting payment networks and distributed ledgers. Potentially, it is an alternative to Swift [46].

Rising to the challenge, SWIFT just added some modern technology components to its messaging network, providing a three-phase set of services called *Global Payments Innovation* (GPI). Its first phase is a FedEx-style tracker for international payments. It provides API access to real-time data about the status of an international payment as it passes through each bank in a correspondent banking chain. In the second phase GPI will introduce a tool to stop immediately a payment, and the option to transfer data for compliance checks within payments. Eventually, in the third phase it will support blockchain technology for *Nostro* reconciliation, where banks hold foreign currency in accounts at other banks to facilitate foreign exchange and trade transactions.

3. Concepts and Technologies

As we saw in previous sections, during the last years a new generation of systems built on concepts from Bitcoin were emerging. Although these systems can markedly differ in scope, deployment scale and/or underlying technologies, they all stem from the Bitcoin original idea of managing peer's interactions through a tamper-proof, time-oriented distributed chain of transaction records, that is the *blockchain*.

In this section we will try to describe the main technical aspects at the basis of blockchain-based systems, at the current state of the art. Before going into details it can be useful to recap Bitcoin's basic strategy by sketching its main operations. This can serve as a guideline before our diving into the technicalities, and we hope it will induce the reader to focus on the functional requirements pursued by the designers of these systems, rather than on their actual implementations. For an in-depth technical overview explicitly focused on cryptocurrencies the reader is referred to [47].

Table 1 reports the main special symbols used throughout the following, together with their meaning.

Table 1. Special symbols used in the present work.

Symbol [Section(s)]	Meaning [Section(s)]	Symbol [Section(s)]	Meaning [Section(s)]
T [3.2.2]	Transaction [3.1,3.2]	M [3.3.1]	Miner [3.1]
I [3.2.2]	Transaction input [3.2.2]	R [3.2.3]	Recipient (payee in cryptocurrency) [3.2]
O [3.2.2]	Transaction output [3.2.2]	P [3.2]	Transactor (payer in cryptocurrency) [3.2]
B [3.3.1]	Transaction block [3.3]	N [3.3.2]	Solution of mining (nonce) [3.3.2]
H [3.3.1]	Block header [3.3.1]	τ [3.3.2]	Difficulty of mining (threshold, aka target in Bitcoin) [3.3.2]
D [3.3.1]	Digest [3.2.1]	Ψ [3.3.2]	Pseudo-random function [3.3.2]
$\widehat{D}$ [3.3.2]	Root hash (aka root digest) [3.3.2]	Π [3.3.2]	Pricing function [3.3.2]

3.1. Bitcoin Quick Overview

One main problem with digital currencies is that of *double spending*, where the same coin is transferred by a given payer to two or more payees. In the traditional model, double spending is prevented by trusted parties (i.e., banks and other financial institutions), which interoperate to enforce a total order in the processing of coin transfers and their balancing over time.

Bitcoin raises every user to the role of a central bank, and tackles the double spending threat by fostering a competition among them. Indeed, every participant in the Bitcoin network:

- keeps a copy of the ledger recording transfers related to digital coins/assets over time;
- performs a certain amount of publicly-verifiable computational work in order to have a chance to be rewarded in new mint coins plus some fees by the payers;
- checks the work performed by other participants in order to get more chances to be rewarded in the future.

In the Bitcoin jargon, users acting as central banks are *miners*, transfers of digital assets are *transactions*, and the publicly available shared ledger where transactions are recorded is the *blockchain*.

In an ideal Bitcoin network the blockchain is cooperatively and incrementally constructed by the miners over time as follows:

- Miners collect transactions that are broadcast over the network, and use their computing power to try to generate a valid block of transactions. The generation is done through repeated invocation of a hash function on data which reference the specific transactions that a miner decided to include in its block, together with the previous valid block and its own Bitcoin address. (Since Bitcoin inception, the hash function has been implemented with a double call to the SHA-256 algorithm [48]. However, as the requirement of collision resistance is quickly hardening with the diffusion of Bitcoin and the introduction of the new hash standard SHA-3 [49], this could change in a near future.)
- When a miner succeeds in generating a block, meaning that the hash of its block data is smaller than a given difficulty threshold (aka *target*), then it broadcasts such block to the network.
- In case other miners see that the above block is valid, and see that it is the "longest extension" of the blockchain (i.e., the fork involving the highest amount of computational effort) that they are aware of, they move on and extend the blockchain from such block.
- The existence of the above block in the blockchain ratifies that the newly minted coins and the fees from the transactions included go to the public address that it provided. Only the miner having that address can redeem such coins and transaction fees, by using its corresponding private key.

At the same time, the synchronization among peers and the creation of new coins are managed by the system in a way that:

- The difficulty level readapts to the total hashing power of the miners. This is done by updating the hash threshold value every 2016 blocks, so that so many blocks get generated in about two weeks (i.e., each new block is generated every 10 min on average).
- The reward in newly minted coins started at 50 coins in January 2009 and halves every 210,000 blocks, i.e., about every 4 years (see above).

According to the last rule, the amount of minted coins for each block halved for a first time on 28 November 2012 and for a second time on 9 July 2016. Assuming the Bitcoin system will survive as it is in the next three years, the next halving will occur on 19 July 2020, resulting in 6.25 new minted coins of reward.

As for the difficulty level, the total hashing power of the miners has followed until now an almost continuous exponential growing trend, with a floating point representation for the difficulty (`bdiff`) that from an initial value `bdiff` $= 1$ (corresponding to the easiest target) has reached the value `bdiff` $= 708659466230.332$ on 13 July 2017. These and other statistics about the Bitcoin blockchain are easily available on-line through a *blockchain explorer*, a web service designed to offer human-readable views of a blockchain (e.g., [50,51]) or some other related services (e.g., [52]). Please, see Section 3.3.2 for details on how difficulty is computed and Section 4.1 for some more recent statistics on the Bitcoin network and their implications.

3.2. Transactions

At its core, a blockchain-based cryptocurrency is a chain of digital signatures reflecting the coin's paths through the system. A key concept in this context is that of *transaction*, which has an effect similar to a wire transfer between bank accounts. Actually it is a data structure which represents a single addressable unit of a *block*, which in turn is the single addressable unit of the blockchain. There are two different types of transactions: *standard* transactions, which encode one or more coin's transfers from some *transactor* node to one or more recipient nodes of the system, and *generation* (aka *coinbase*) transactions, where newly generated coins are granted by the system to miners, as a reward for their effort in building the blockchain.

Every amount of currency which results from a transaction is seen as a pack of coins which can be fractionated only through another transaction. Indeed a standard transaction does not record a single fund transfer from one point to another, but rather represents a scheme with multiple amounts which must be balanced: the number of coins-possibly collected from different packs-representing the requested amount for a given transaction must match the number of coins made available for the possible multiple recipients. In fact also the transactor itself can be considered the recipient of the eventual change, which represents a new smaller pack of coins.

This way parties can more easily keep track of paths followed by coins through the system, and monitor the integrity of the service. Actually, many systems include (optional) *transaction fees* for miners, so that the above balance is achieved by including these fees, even if their payee is not known beforehand. Indeed, a transfer for a transaction fee cannot be explicitly recorded in a standard transaction, since its recipient will be known to the system only afterward, resulting as the winner of a computational challenge (see Section 3.3). Therefore, these fees must be included in the amount of the transaction, but they are not explicitly recorded in the transaction itself.

The specification of a transaction requires an *addressing scheme* so that transactors, recipients and miners (that is, ultimately, each node in the system) can be univocally identified. Moreover, transactions must be uniquely binded to transactors, in a way that-at least in theory-neither a party different from the transactor P can generate a transaction indicating P as transactor (*authenticity*), nor P can repudiate a transaction which it has previously generated (*non-repudiation*).

Transactions are processed locally by each node thanks to a programmable interface implemented in the peer-to-peer software. Since that interface makes use of a scripting language (e.g., [22,53]), the code for processing a transaction is referred as the *transaction script*. The ultimate goal of a transaction script is allowing any other node in the system different from the transactor node to ascertain what are the conditions for the occurrence of transfers reported in the given transaction. That is accomplished through the signing and the verification procedures of a suitable digital signature scheme on some elements of the transaction.

These notions, related to transfers of electronic funds, were first introduced in Bitcoin, and then adopted by many other cryptocurrency systems to date. They can be easily adapted to perform transfers of any resource which admits a suitable, appropriate digital representation. Thus, by definition, we will assume that this approach specifies how resources are managed in any blockchain-based system.

3.2.1. Addressing

Since the validation of transactions relies upon digital signatures verified by system nodes, and nodes are implemented through a special software termed *wallet*, then there must be a mapping between wallets and (sets of) signature verification keys.

Signature verification keys are often linked to identifiers or pseudonyms valid only in the system context, called *addresses*, rather than to real-life names and identities as in infrastructures based on public-key certificate (PKC) [54]. This way, no trusted third party is required to ascertain the real identities of people and devices involved in the system; moreover, users can hope for some anonymity.

For example, Bitcoin addresses are base58-encoded strings containing: the address version; the digest got by hashing an ECDSA [55] verification key twice, firstly with the SHA-256 algorithm [48] and secondly with the RIPEMD-160 algorithm [56]; and an error-detection checksum to catch typos [57]. It is controversial if, by introducing addresses, the Bitcoin's designers aimed at some obfuscation of the verification keys, besides obtaining shorter identifiers. Indeed, verification keys must be ultimately involved in the verification process, and can be used by themselves to address transaction's endpoints. Actually this is the case for Bitcoin, where addresses are used to receive coins and their related verification keys are used to redeem them.

In any case, the approach outlined above is inappropriate if a non-volatile addressing is required, since a secure key management imposes practices such as *key upgrading* and *key revocation* [58]. But in cryptocurrencies volatile addresses can be valuable, since they can appreciably improve the degree of anonymity. Users' anonymity is a feature aimed by those cryptocurrencies, like Bitcoin, that intend to emulate cash money, in contrast to the ever-growing tracking capabilities associated with other methods of digital payment. With a large pool of verification keys associated with the same wallet, that is using many different addresses (ideally, a new address for each new transaction), an user can hope to conceal her trading through the system. This enforces also some security in key management, because of lower exposition to comparison-based attacks on digital signatures [59].

However, blockchain analysis techniques are available to threaten such approach to user anonymity by tracking coin flows [60,61] and correlating IP addresses to transactions [62]. Conversely, these techniques can be thwarted only by decoupling the information about the transactor and the receiver, in analogy to mix networks like Tor [63].

3.2.2. Transaction Inputs and Outputs

As we told previously, the scope of a transaction is to keep track of the resource paths (both the granting of new resources and resource transfers) occurring among system nodes. In this way each node can verify the amount and destination of the newly created resources, as well as the balance between resources requested for the transaction and resources transferred to recipients. In order to accomplish that, a transaction encodes those amounts as suitable data structures, called transaction inputs and outputs, respectively.

A *transaction input* composes of data indicating the source of the resources that a transactor P wishes to transfer, plus a scripting code whose processing should guarantee to every node in the system that P is actually authorized to redeem those resources. Similarly, a *transaction output* composes of the indication of the destination where P intends to transfer those resources and their amount, plus a scripting code whose processing should guarantee that only the recipients indicated by P can redeem such resources.

Each input must match with an output related to a previous transaction which was validated (eventually among others) as a blockchain block. The matching between the current transaction input I and a previous transaction output O transforms *all* the resources provided by O in resources available for I, so that to each input can correspond one and only one output. A different situation occurs inside a transaction, where zero to many inputs can be associated to one or more outputs. A no-input transaction stands for granting some newly created resources to one or more outputs, and corresponds to the case of a generation transaction in the context of cryptocurrecies. Otherwise, a single-input or multi-input transaction records a transfer of resources from its input(s) to its output(s). The matching rule in this last case is as follows: one or more outputs O_{mn} must correspond to each input I_m, so that a certain amount of units of resource provided by I_m is equal to the sum of the amounts related to O_{mn}, net of transaction fees which cannot be indicated explicitly.

As result of the two above matching rules, resource paths start at outputs of generation transactions and from standard transaction inputs they branch into one or more outputs, or from multiple transaction inputs they join into a single output. Figure 1 shows an example of a chain graph for resource transfers which meets such rules.

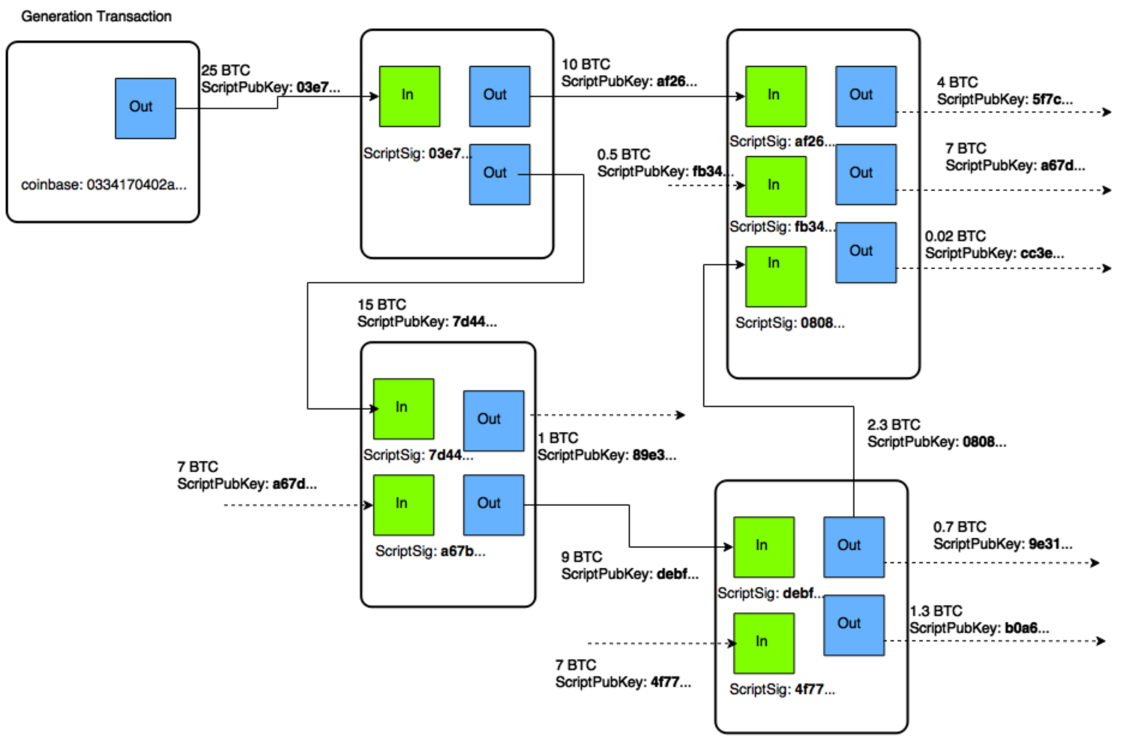

Figure 1. An example of transaction chaining in Bitcoin. For readability, `In` and `Out` matching scripts are indicated with the same Bitcoin public key, but actually each `Out` script encodes the Bitcoin address corresponding to the public key in its matching `In` script. See Section 3.2.3 for details.

In Bitcoin, each standard transaction—with its input(s)-output(s) matching—represents a transfer of a certain amount of *sathoshi*, which is the smallest possible unit of currency (a satoshi is a one hundred millionth of a single bitcoin (BTC)). Each input I must reference a single output O from a transaction encoded in a previously validated transaction block, so that I must spend the amounts of satoshi in O. Schematically, given a transaction T, each input I and output O of T are defined as:

$$I := (\text{sha256}(T_p), i, \text{SgnCode}) \quad O := (v, \text{VrfCode}) \,, \tag{1}$$

where:

- sha256(T_p) is the SHA-256 digest of a previous transaction T_p. It is used to identify T_p as the transaction used to get input I;
- the *input index* i is a non negative integer which indicates the specific output in T_p used as an input in the current transaction T;
- the *signing script* SgnCode is a set of instructions and data provided by the transactor in order to satisfy the conditions placed in the pubkey script of output i in T_p. In its simplest form, such script just applies the transactor signature on the SHA-256 digest of all the I/O fields of the current transaction, with the exclusion of SgnCode itself;
- the *output value* v is the number of satoshis associated with O;
- the *pubkey script* VrfCode is a set of instructions and data provided by the transactor that specifies the recipient(s) authorized to collect the value v in satoshi associated with O. The most used VrfCode returns true if matched with a signing script consisting in a signature of a (subsequent) transaction (see Section 3.2.3).

An example of Bitcoin standard transaction is given in Figure 2. The transaction, identified by the SHA-256 hash indicated at the top of the frame and listed in line 2 of the code as `"txid"`, takes a single input I (coded in the `"in"` section) and converts it in the two outputs O listed in section `"out"`. I is identified by the `txid` and `n` fields given in lines 9-10, which correspond to the SHA-256 digest and index of a previous Bitcoin transaction. The first output transfers a `value` of $v = 12134079$ satoshi to the Bitcoin address given by `1HJjNPksrnqW374Hb7GhtaKUysDQcvnF1K`, whilst the second output transfers $v = 5933405$ satoshi to the address `1GKQwKx54ahKQFQ4rya54nG5f9w1C8pCYt`. Notice that this is a real transaction, relayed by IP 46.165.220.71 (whois) and included in the block number 477213 of the Bitcoin blockchain on 23 July 2017. The next section explains how this transaction was created by its transactor, and the way Bitcoin processed it.

3.2.3. Transaction Processing

In order to be validated without the intervention of a trusted third party, transactions require some processing at peer nodes to ascertain with corroborate evidence: their origin (i.e., the identities of their transactors), the fact that the resources deemed to be transacted are effectively possessed by the transactors, and—last but not least—that only the recipients satisfying the conditions required by transactors are accounted for such resources.

A key point in Bitcoin, and more generally in blockchain-based systems, is that transactions include code that can be customized by the transactor in order to allow for different kinds of verification procedures. To verify that inputs are authorized to collect the values of referenced outputs, Bitcoin uses a scripting system named Script [53] which resembles to Forth [64]. With reference to the transaction T of Figure 2, the input's SgnCode (denoted as `scriptSig` in the code) and the referenced output's VrfCode (`scriptPubKey`) are evaluated, and the input is authorized only if `scriptPubKey` returns true. In this case the transactor P decided to transfer a total amount of 0.18067484 BTC from a previous transaction output given by `(txid=fa53f5..., n=1)` to the two recipient addresses `1HJjNP...` and `1GKQwK...` and, since the output's value was of 0.1824822 BTC (this in not indicated in T, and is desumed by inspecting the value associated with output `n=1` of `txid=fa53f5...`), P payed 0.00180736 BTC fees to get T processed.

T: 706b335514c44e6311439db679f049788122689a21bd40a1b73306c8ff33f894

```
{
"txid": "706b335514c44e6311439db679f049788122689a21bd40a1b73306c8ff33f894",
"size": 225,
"version": 2,
"locktime": 0,

"in": [
{
"txid": "fa53f5c75a28274f3ae029238b4dd03daf2adbaf79bdf221c925958a7c3d8a27",
"n": 1,
"scriptSig": "304402201730eba212e9c8370b9aa571ca184d3c31d4aa329b840291227e96bd620b687c02204572c0ea5bf529781f8f1d
3d53e5263a5ebb784302fb7d638d1b2452709f95f001 02e13d9fab3c4730183d14208163e532de1b61e9b79c8b102a686ac0b8e57b076d"
}
],
"out": [
{
"value": "0.12134079",
"n": 0,
"scriptPubKey": "OP_DUP OP_HASH160 1HJjNPksrnqW374Hb7GhtaKUysDQcvnF1K OP_EQUALVERIFY OP_CHECKSIG",
},
{
"value": "0.05933405",
"n": 1,
"scriptPubKey": "OP_DUP OP_HASH160 1GKQwKx54ahKQFQ4rya54nG5f9w1C8pCYt OP_EQUALVERIFY OP_CHECKSIG"
}
],
}
```

Figure 2. An example of Bitcoin standard transaction.

The flow of operations performed by P to encode T can be described as follows [57]:

1. the `out` section of T is created by instantiating the two `value` fields and by putting the addresses `1HJjNP...` and `1GKQwK...` after the `OP_HASH160` instruction in their related `scriptPubKey` fields;
2. the previous output `(txid=fa53f5...,n=1)` is indicated as the source of the input in the `in` section;
3. P completes the encoding of T by writing in the `scriptSig` field its signature `304402...` followed by a space and its public key `02e13d...`. It can be worthwhile to notice that P's signature is applied to all the `in` and `out` fields of T, with the only exception of `scriptSig`.

In order to better understand the processing of T by the Bitcoin scripting system, it is useful to think of T along with transaction T_p referenced by `txid=fa53f5...` and instantiated by P_p, and a subsequent transaction T_s which the recipient R, having address `1GKQwK...`, issued to redeem the 0.05933405 BTC (at the time of writing the output associated with address `1HJjNP...` is still unspent).

Indeed, actually two computations took place: first the `scriptSig` of T was given in input to the `scriptPubKey` provided in output `n=1` of T_p, and then the `scriptSig` produced by R for a given input of T_s was given as input to the last `scriptPubKey` of T (lines 22–24 of the code in Figure 2). In both cases the processing was successful; it is described in Table 2, where the values `<sig>`, `<pubkey>` and `<pubKeyHash>` are respectively instantiated with P_p's signature, P_p's public key and P's address in the first case, and P's signature, P's public key and R's address in the second case.

Table 2. The stack-oriented processing of Script in case of "Pay-to-PubKeyHash" (P2PKH) scripts. Symbology is according to [57].

	Step 1	**Step 2**	**Step 3**	**Step 4**	**Step 5**	**Step 6**	**Step 7**	
Processed inst/data	`<sig>`	`<pubkey>`	`OP_DUP`	`OP_HASH160`	`<pubKeyHash>`	`OP_EQUALVERIFY`	`OP_CHECKSIG`	
Stack content					`<pubKeyHash>`			
			`<pubkey>`	`<pubKeyHash>`	`<pubKeyHash>`			
		`<pubkey>`	`<pubkey>`	`<pubkey>`	`<pubkey>`	`<pubkey>`	`<pubkey>`	
	`<sig>`	`<sig>`	`<sig>`	`<sig>`	`<sig>`	`<sig>`	`<sig>`	`True`

What described above is a so-called "Pay-to-PubKeyHash" (P2PKH) script, currently the most used way in Bitcoin to transfer coins from one or more origins to a destination. At the moment, Bitcoin provides also "Pay-to-Script-Hash" (P2SH) scripts, introduced with the motivation of moving the responsibility for supplying the conditions to redeem a transaction from the transactor to the receiver. P2SH scripts provides a means for encoding arbitrary transactions, i.e., smart contracts, since in this case `scriptSig` encodes a signature on a generic script (which however must have a prescribed serial format in order to be successfully interpreted by Script), whilst `scriptPubKey` ensures that the script supplied in the redeeming transaction univocally corresponds to the script used to create the address [53]. Script allows the implementation of many different conditions that peers have to meet in order to claim the output's value, like (refundable) deposits, *assurance contracts*, *pay-for-proof* contracts and rapidly-adjusted (micro)payments to a pre-determined party [65].

A similar approach is used in other blockchain-based systems to date, although some differences exist in the scripting system used, and in the way transaction processing takes place.

Particularly relevant seems in this respect Ethereum (see Section 2.2), which extends Bitcoin's transaction programming by introducing conditional loops through its virtual machine EVM. Loop's control statement are deliberately omitted in Bitcoin to avoid deadlocks during transaction verification. Because of the stack-based nature of Script, that results in a *not Turing-complete* instruction set, which was pointed out by Buterin in [26] as a major limitation to the creation of complex transaction scripts required for "colored coins" (i.e., custom currencies and financial instruments) and smart properties, or of even more complex scripts for smart contracts and DAOs. These, on the other hand, represent Ethereum's core objectives.

The EVM has its own low-level language (bytecode), but high-level programming languages have been developed from the launch of the Ethereum platform, in order to facilitate the creation of contracts. The most commonly used and supported of them is probably Solidity [22], which looks like Javascript. Another notable scripting language for writing Ethereum contracts is Serpent [21], which was designed to be very similar to Python. Any code written in these languages gets compiled into the the EVM bytecode.

Ethereum uses its own digital coin, called *ether*, to "fuel" the execution of applications stored on its blockchain. *Fueling* is designed as a way to pay for the execution of Ethereum bytecode and storage of data on the blockchain; notably, it represents also a countermeasure to indefinite looping, a threat to which are exposed Turing-complete systems. If a transaction execution "runs out of gas", then all state changes revert, except for the payment of the fees; conversely, if the execution halts with some gas remaining, then the remaining portion of the fees is refunded to the transactor.

How much a specific transaction will cost in Ethereum is defined by the complexity of its computation in term of some given (sets of) operations, whose relative costs are expressed in terms of "units of gas". In turn, a unit of gas is priced `GASPRICE`, a value representing the fee that the transactor has to pays per computational step. The `GASPRICE` value–along with `STARTGAS`, which represents the maximum number of computational steps the transaction execution is allowed to take–are the two parameters introduced in [26] to set up the Ethereum's anti-denial-of-service mechanism. This mechanism has changed a bit during the development of Ethereum, perhaps also in the attempt to overcome the issues sketched in Section 2.2. What is currently considered by the community the reference (but rather illegible) document for Ethereum's implementation specifications [66]—aka "The Yellow Paper"—describes a more detailed mechanism and reports in Appendix G a fee schedule in term of units of gas for 31 abstract operations that can affect a transaction. This last pricing mechanism seems quite cumbersome and it will be hopefully simplified, along with some other convoluted specifications, in a future release of Ethereum. Actually, some little discrepancies exist between [26] and [66] also in term of parameter naming and meaning. For example, `STARTGAS` has been replaced by `gasLimit`, which represents the maximum amount of gas that should be used in executing a transaction. Moreover, `GASPRICE` (`gasPrice`) is now explicitly indicated as a value which can be freely specified by transactors but that will be necessarily subjected to a trade-off with miners between lowering the gas price and maximising the chance that their transaction will be mined in a timely manner.

However, beyond details and speculations concerning current and future developments, Ethereum roots on: (i) the assumption that Turing completeness is required to built complex smart contracts and, (ii) the idea that the fuelling mechanism can help the application developers to face attacks which exploit in some way the greater computational power offered by Turing completeness. Unfortunately, both (i) and (ii) seem controversial.

3.3. The Blockchain Technology

As we have seen in section Section 3.2, blockchain-based systems use digital signatures in order to protect *single* transactions from tampering, so to guarantee the proper attribution. However, one main concern for such systems is that of protecting from tampering a chain of transactions like that depicted in Figure 1, and ultimately all the chains of transactions starting from the *genesis block*, which is a special case of generation transaction where a given amount of coin is appointed to one or more users in order to start the system. The problem arises since such chains are built incrementally by a network of peer nodes, without the support of any trusted third party. Actually, neither miners nor transactors can be trusted, since each of them could cheat in order to get more resources or to avoid that other parties get the resources they deserve.

Suppose, just to fix the ideas, that the system goal has been to manage payments for the purchase of goods within a set of parties, where each party can play both the roles of buyer and seller. This is similar to the application scenario for cryptocurrencies, but without the basic issue of coining new currency without the aid of a trusted third party; in other words, we are considering the case where digital money is issued by a trusted authority, and the system only manages money transfers. Then, obviously, the sellers want to be assured that payments that were agreed with the buyers–as recorded in the transactions–cannot be declined by the buyers themselves at a later time, e.g., when goods have already been shipped. One main question arises in this respect: Can a chain of digital signatures protect sellers from buyers who cheat without the aid of a trusted authority?

The answer is affirmative, even if this solution does not seem to have been implemented. Indeed a seller R can protect herself from attempts to alter a transaction T_P in her favor if she creates a subsequent transaction T_R which redeems per se the money provided by T_P. Therefore, the only way for P to alter T_P is to collude with R, since only R can generate the input of T_R, and such input must match with the output of T_P.

Thus, the chains of standard transactions illustrated in Section 3.2 can protect against the so-called *double-spending problem*, where malicious users try to transact some resources (the same digital coins, in the context of cryptocurrencies) twice or more. However, our assumption implies that money cannot be cloned, since it is provided by a trusted party which authenticates each single issued digital coin thanks again to a digital signature and a serial number.

Things get more complicated if the system has to provide for the creation of new resources (e.g., the coining of money) without a trusted authority; in this case, both transaction chain tampering and resource forging are possible threats. Moreover, these other two main questions arise: (i) To which node(s) should in this case the new resources be granted?, and (ii) How to account such grants in the system?

The idea in Bitcoin was to assign the task of assessing the correct matching of standard transactions to special nodes called *miners*, to empower them for the tamper-proof resistance of all the transaction chains, and to reward them for their work with some (optional) transaction fees plus new generated coins. Thus, each miner must associate a *generation transaction* to the processed standard ones. Moreover, since grouping several standard transactions together helps restraining network overhead, a miner can refer to a single generation transaction when processing a group, and eventually she will be awarded with newly generated coins and relative *transaction fees*(see Section 3.2).

Following this idea, the "unit of mining" becomes the *transaction block*, a data structure grouping one or many standard transactions (by one or more transactors) with a single generation transaction. This is where firstly the blockchain technology comes into play. To put it simply, the blockchain technology consists of a data structure (the *blockchain*), plus a protocol for managing the creation of new transaction blocks. As previously stated, the goal pursued with this technology is to have a public ledger of transactions built over time thanks to the miners, and whose consistency and authenticity can be monitored by any node in the peer-to-peer system, without any central authority.

However, the shift from single transactions to transaction blocks exposes again the system to transaction chain tampering. Indeed, in the context of an organization of recorded data in transaction blocks, chains of transfer paths cannot be easily managed and validated through digital signatures as described in Section 3.2. To overcome this difficulty, the designer(s) of Bitcoin introduced a new chaining mechanism, in this case between different blocks. Roughly speaking, it consists in the insertion of a cryptographic hash digest related to the preceding block in the current block. Since a cryptographic hash digest is supposed to correspond univocally to its originating data with overwhelming probability, then the above approach results in the circumstance that any tampering in a given block involves changes in all its subsequent blocks.

But the above chaining of blocks by itself cannot solve the replacement of a tail of the blockchain with another maliciously chained tail. Indeed, such replacements can involve the cloning of money (i.e., double-spending), since money is issued through generation transactions. And this is where mining comes into play: by making the mining of a transaction block a costly computation, block replacement is made difficult, and the double-spending threat is mitigated. Indeed, in order to achieve its purpose, a malicious miner has to be faster than the honest miners in mining transaction blocks.

Summing up, the management of the blockchain—that is, the collection of new transactions and their insertion upon validation in the chain of previous blocks—is a process composed of the following steps:

1. Collecting incoming transactions into new blocks;
2. Mining of new blocks;
3. Blockchain updating.

All the above steps are performed by miners thanks to suitable client software interacting with their *wallets* (see Section 3.2.1). This software enforces protocol executions using cryptographic mechanisms as preventive and detective controls, and implements the decision-making process required at step 3 through a kind of consensus expressed by the nodes involved in the system. The following sections discuss each step in detail. A schematic representation of the main elements composing the Bitcoin blockchain and their processing is given in Figure 3.

3.3.1. Collecting New Transactions into Blocks

It's a miner task to collect new transactions that are taking place on the network, and to assemble these transactions into a transaction block. In Bitcoin and other cryptocurrencies, miners are totally free to choose which transactions to include in their blocks. Most miners will consider any transaction that reaches them over the network, assuming it includes an appropriate fee, but nothing forces their choice. This freedom is intended to get resilience and performance of the system in case of network delays or faults, since new transaction broadcasts do not necessarily need to reach all miners, nor a miner has to wait for a minimum of transactions in order to start on working on a new block [7,57]. Actually, it could be also the case that a miner with malevolent intentions will decide to mine a block composed of just the generation transaction, with the purpose of having a greater chance of getting that block inserted in the blockchain. As we will see later, there are very few deterrents to a such bad behavior, that would eventually result in lack of faith in the system (see Section 3.3.3).

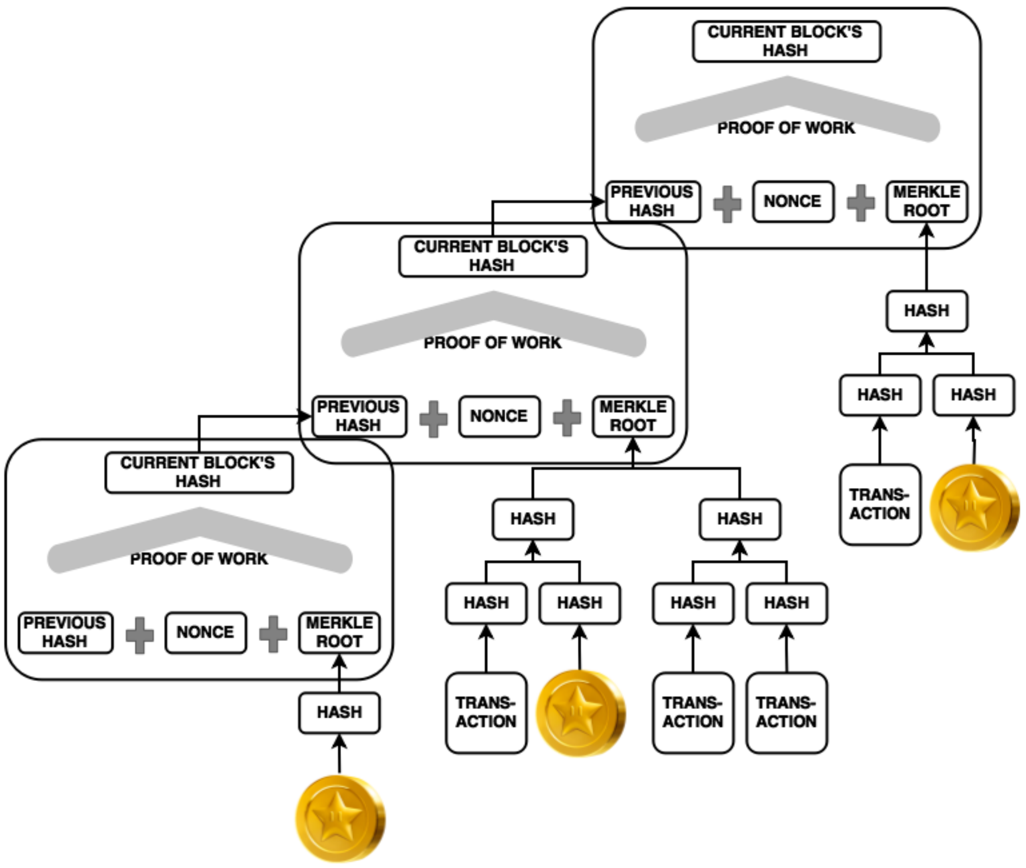

Figure 3. The core elements of a blockchain like that of Bitcoin, with *generation* transactions depicted as coins. The block on the left could be the *genesis block* or a block generated by a selfish miner as well. See Section 3.3.1 for details.

There is, however, an upper limit to the size of a transaction block, which is currently 1MB in the Bitcoin system. Such value, which has been the subject of a long debate in the Bitcoin community and will probably change in the next Bitcoin release [67], has to be chosen as a trade-off between constraints on peers resources and the throughput of the transaction management: a bigger limit would result in less nodes being able to mine or validate a full-size block, whilst a smaller limit would imply less transaction throughput [68].

In Bitcoin, the standard transactions $T_1, \dots, T_h$ that a miner M chooses to assemble into the current mining block B_c—alongside with the generation transaction T_0—are processed by M in pairs (if $h > 0$ and $h + 1$ is odd, then T_h is considered twice), in order to calculate a unique hash digest $\widehat{D}_c$ for all of them (*root hash*) thanks to a *Merkle tree* [69]. The digests used to generate the Merkle tree are all computed by applying the SHA-256 hash function twice, thus $\widehat{D}_c$ is a 256-bit string. The root hash is then recorded in the so-called *block header* H_c of B_c, alongside with the digest D_p of the block header H_p of the previous (valid) block B_p , where D_p is computed by applying twice SHA-256 to H_p. Besides the 256-bit fields provided for storing D_p and $\widehat{D}_c$, a block header consists of the 32-bit fields `version`, `time`, `bits` and `nonce` for a total of 80 bytes. The first three fields store the block version number, the current time in seconds since 1970-01-01T00:00 UTC and a string encoding the difficulty of the proof-of-work, respectively. Instead, the `nonce` field is used to store (part of) a string representing the solution given by M to the proof-of-work (see Section 3.3.2) for B_c [70,71].

Notice that if a miner generates a block with one or more invalid transactions, whatever the error and the intentions are, then the other (honest) miners will not accept the new block. They will ignore it and continue trying to build a block on top of the last block they think is valid.

3.3.2. Mining of New Blocks

In one form or another, *mining* is currently at the heart of any working blockchain-based system. Mining is indeed the way the system imposes a computational cost for the recording of new transaction blocks into the blockchain, so to mitigate the double-spending threat. An important point to stress here, since often misunderstood, is that the goal of a mining protocol does not actually consist in the determination of the miner in charge of block recording, but rather of the set of miners that are eligible for recording a new block in the blockchain (see Section 3.3.3).

The mining protocol chosen in Bitcoin is similar to the (*partial inversion*) *proof-of-work* (PoW) as described in [72], even if in this meaning it has new characteristics and purpose.

In a proof-of-work, a prover demonstrates to a verifier that she has performed a computational work in order to gain access to a resource. The original application for this type of protocols was in the context of being able to deter spam email, and the main idea was to require the prover to solve a moderately hard but easy to verify computational problem tied to a particular email message [73]. That is conceptually like affixing a postage stamp to a message: for a legitimate sender who is only sending out a small number of messages such type of proof will not amount to very much computational effort, but for a spammer it might be prohibitively expensive.

The solution N of the proof-of-work is called *nonce*; the term was inspired by [74] and refers to the fact that, because of the properties of a cryptographic hash function, N behaves as a pseudo-random string as one or more of the proof-of-work's parameters change.

The proof-of-work implemented in Bitcoin consists in the search for a peculiar hash digest D, calculated applying two subsequent SHA-256 to H, such that

$$D = \mathrm{sha256}(\mathrm{sha256}(H)) \leq \tau\,, \tag{2}$$

where τ is fixed by the system in a way that we will see later in this discussion, and H is the header of the block of transactions collected by miner M. As seen in Section 3.3.1, H consists of: the same kind of double-hash digest D_p of the previous valid block header H_p, the root hash $\widehat{D}$ of the transactions collected by M in the current working block B, the `nonce` field, and other minor fields. The `nonce` field is 32-bit long and is used for storing (the first part of) a string N which the miner has to find in order to satisfy condition (2).

With the introduction of special purpose hashing hardware (see Section 4.1) the `nonce` field in the header has become too small, and the workaround was to have an `extraNonce` field in a generation transaction so that a solution N has to be found in a much larger string space. This makes the proof-of-work more costly and involved, since M must recompute the root digest $\widehat{D}$ each time it changes the `extraNonce` string [71].

A miner M to succeed in the search for a D satisfying (2) will write N in the `nonce` and `extraNonce` fields of its working block B, and will broadcast B to the other peers for verification. The other miners in the system express their acceptance of the block by starting to work on the creation of a next block in the chain, therefore they use the digest D of the accepted block as one component of the block header of the new working block, as previously described.

Now the point is to understand how τ is fixed. Practically, the goal of finding a $D < \tau$ corresponds to the goal of finding a D such that it has a certain number of zeroes as first Hex digits. A SHA-256 digest is always 32 bytes or 64 hexadecimals long, so the probabilistic difficulty of the goal increases as this number of Hex zero digits increases, and therefore it increases inversely as τ decreases.

For example, if we need digests beginning with fifteen zeroes like:

$$000000000000000 1b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4,$$

then τ will be the integer number whose Hex representation is given by:

$$0000000000000010000000000000000000000000000000000000000000000000.$$

More generally, if we need digests having z leading zeros, then τ will be the integer coded by the Hex string of all zeros except the digit "1" in the z-th position. Since the threshold τ exponentially decreases over time, with the consequent introduction of the `extraNonce` field in generation transactions, mining has become a very difficult problem which at current time is typically solved by mining pools, where a bunch of miners share work and rewards. Mining pools use higher thresholds than the target threshold τ to see how much work miners are doing. For example, if mining requires a digest having 15 leading zeros, then the mining pool can ask for digests having (at least) 10 leading zeros. Each partial solution proves that the miner is working hard on the problem, and gives the miner a share in the final reward, if any. Eventually one of these partial solutions could get the target number of leading zeros, therefore successfully mining the block and winning the reward for the pool. The reward is then split basing on the fraction of total shares that each miner counts, and the pool operator takes a small percentage for her work [71].

As we are going to better explain in Section 4, Bitcoin's proof-of-work is a way to get a valid and reliable transaction processing by ensuring that a set of peer nodes (the miners) compete in solving a computational challenge within a preset time. However, the problem to be solved is a kind of artificial puzzle of no practical concern (besides, of course, the existence of the network), which requires an increasing amount of computational resources over time, and where miners' fitness is proportional to their hashing rate. In an attempt to overcome these limitations, some alternatives for implementing a requirement like (2) have been proposed over time, concerning both the underlying function used to compute D and the way τ is chosen. These alternatives were introduced mainly in an attempt to mitigate threats deriving from accumulators of hashing power. A single party, or a *pool* of them, could indeed invest in hardware equipment to obtain a substantial percentage of the total hashing power, and with such power they could succeed in double-spending by reversing the recent blockchain history. Or alternatively they could carry out a denial-of-service by refusing to include transactions in the block they generate, unless perhaps the transactions useful to their own purposes.

Although only a percentage greater than 50% of the total hashpower allows an attacker to get the mathematical certainty in subverting sooner or later the system [7], also attackers with a substantial minority of the total hashpower can plausibly attempt double-spending and denial-of-service attacks [7,75].

Litecoin [76] makes use of a variant of Bitcoin's proof-of-work that employs sCrypt [77] instead of SHA-256. sCrypt was originally designed as an alternative to the iterative use of pseudo-random functions like MD5 [78] or DES [79] for deriving a key from a password/passphrase. Indeed sCrypt's goal was to implement a pseudo-random function Ψ which is also *sequential memory-hard*, i.e., such that:

- the fastest sequential algorithm for computing Ψ requires an amount of memory roughly proportional to the number of operations to be performed;
- it is impossible for a parallel algorithm to asymptotically achieve a significantly lower cost in terms of both cost and time than the fastest sequential algorithm.

This feature is used in Litecoin and a bunch of other cryptocurrencies [10] to hinder the use of specialized hardware, thus avoiding the side effects of the rapid shift in mining technologies observed for Bitcoin. Other interesting derivatives of Bitcoin's proof-of-work, aimed at improving its efficiency or at doing a more useful work, are described in Section 4.

Those cryptocurrencies that do not make use of a proof-of-work commonly rely on the *proof-of-stake* (PoS) approach to mining. These are mechanisms (e.g.,[80–82]) that increase the chances in mining new blocks for the stakeholders of the system, with the aim to get a counterweight to hashpower. In a proof-of-stake mechanism the threshold τ is indeed proportional to the number of coins owned by the miner at current time, meaning that a miner which owns, e.g., 100 coins is 10 times more likely to create the new block than a miner which owns 10 coins. By itself this would result in "coin pile issue" similar to the hashing power trouble that a proof-of-stake is intended to mitigate, thus these mechanisms also include some other parameters to compensate such bad behavior. For example, in the PeerCoin system τ is proportional to *coin age*, which is the number of coins owned times the number of days the coins have been held. An important difference here is that the hashing is timed in one digest calculation per second, and not more frequently; moreover it is possible to calculate it only once for each unspent coin in the wallet [80]. In this way miners are not motivated in accumulating hashing power. However, this comes at the price of a (loose) time synchronization among nodes, which was presumably avoided in Bitcoin to overcome the difficulties of getting a trusted "global clock" without a *Time-Stamping Authority* [83].

In the last two years, proof-of-stake mechanisms have gained approval for their lower energy consumption and greater fairness for the selection of the winning miner with respect to proof-of-work derived systems. Moreover, they are considered more resilient to the double-spending threat. In proof-of-stake systems the "longest chain" is the blockchain branch with the highest total sum of destroyed coin age, thus a 51%–attack (that is, when an attacker has the absolute majority of resources necessary to solve the mining) would require a huge amount of coins by the attacker, who needs to control the network even when the coin age turns to be zero. The idea of ruining a large amount of coins is considered a deterrent for this kind of attacks. However, also the proof-of-stake approach has shown some limitations, e.g., coin age accumulates even when the node is not connected to the network [47].

The *proof-of-activity* (PoA) is one more approach to mining, which tries to overcome the limitations of both proof-of-work and proof-of-stake mechanisms. It can be considered a combination of them, since its basic idea is to draw a fraction of the proof-of-work reward only among all the nodes that are connected to the network, while their stake determines their chances of winning [84].

We conclude this section by observing that, at least at our knowledge, all the mining approaches proposed so far can be described as particular instances of the workflow depicted in Figure 4, here described.

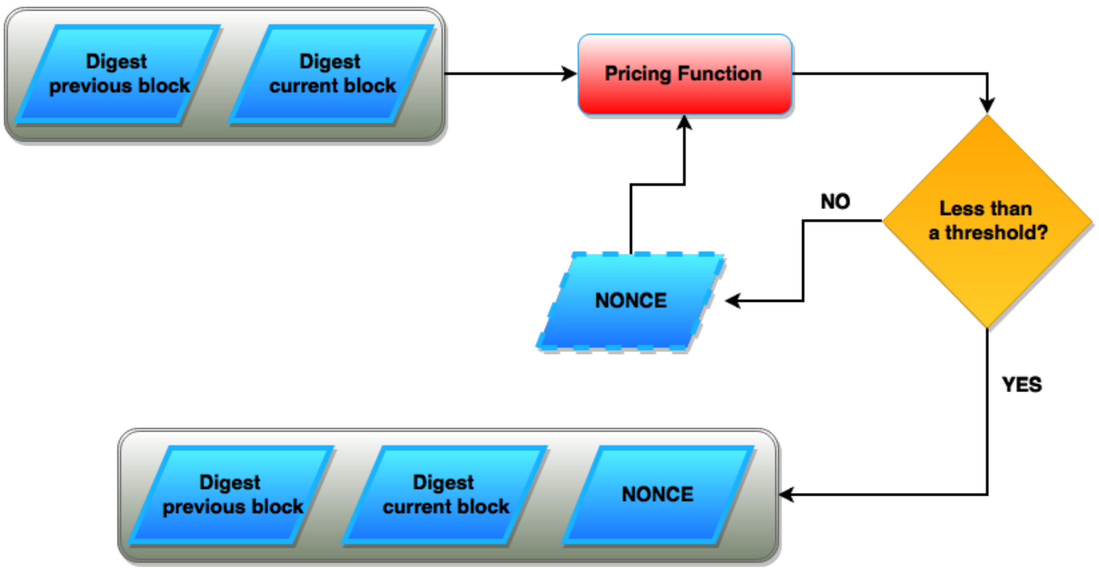

Figure 4. Given the digests of the current and previous blocks, the act of mining the current block is a process which consists in finding a nonce so that the value returned by a suitable pricing function is less than a parametric threshold. The output consisting in the association between the digests and the nonce actually represents what is indicated with "proof-of-work" or similar terms (e.g., "proof-of-stake") in literature.

Pricing functions were introduced in [73], where a function $\Pi : X \times Y \to \mathbb{R}$ can be informally defined as satisfying the following properties:

1. Given $x \in X$ and $\tau \in \mathbb{R}$, it is moderately hard to find $y \in Y : \Pi(x, y) \le \tau$;
2. for any $(x, y) \in X \times Y$, it is easy to compute $\Pi(x, y)$;

3. Π is not amenable to amortization: for any finite set $x_1, \ldots, x_m \in X$ of input values and corresponding threshold values $\tau_1, \ldots, \tau_m \in \mathbb{R}$, the cost of finding $y_1, \ldots, y_m \in Y$ with $\Pi(x_i, y_i) \leq \tau_i$ on the whole set is comparable to the total cost of finding $y_i : \Pi(x_i, y_i) \leq \tau_i$ from scratch for each (x_i, τ_i);

Requirement (3) means that there are no "shortcuts" in finding $y : \Pi(x, y) \leq \tau$ for multiple inputs and thresholds, also if these inputs and thresholds are strictly related. In other words, the solution y of the search problem related to Π behaves like a random variable in Y.

The sense of the workflow of Figure 4 is as follows: through a suitable pricing function Π and a tunable difficulty threshold τ, a miner node M can give corroborate evidence to its peers that it found a solution to a moderately hard computational problem, and that its effort involved processing of the previous block in the blockchain alongside with the current transactions.

3.3.3. Blockchain Updating

As we described in Section 3.2.2 and illustrated through Figure 1, the chaining of a single transaction results in a quite complex graph, where a single input may branch in multiple transaction outputs, and multiple inputs may merge in a single transaction output. Things go better when transactions are assembled in blocks and these blocks are chained together to form the blockchain (see Section 3.3.1). The blockchain has indeed a tree data structure: the genesis block is its root, each block is a child of the block it references, and it may be the case that a given block has more than one child, thus resulting in one or more branches in the chain. That sounds odd, since a single coherent history of transactions is desired, and each branch represents one different version of such history. However, it is consequence of the approach chosen to add a new block in current blockchain-based systems. Following [7], indeed, all these systems adopt an implicitly-defined majority decision, consisting in the fact that (honest) miners will spend their next mining work on the chain of blocks with the most mining effort in it, the so-called "longest chain". Such effort is measured as the sum of the difficulties that were required to mine all the blocks composing the chain [85]. In proof-of-work based systems, for example, the longest chain is the one with the longest string got by concatenating the "leading zeros" of all the proofs of work that were required to build the chain (see Section 3.3.2).

In order to better understand the process of updating a blockchain, it may be worth pointing out here the following circumstances.

- The majority decision mechanism described above is not mandatory, and the choice of the block to be referenced in the chain is actually left to each single miner. A miner M is just encouraged to adhere to the mechanism, on the basis of the fact that the longer is the chain linked to M's new mined block B_M, the higher are the chances that M gets a reward for the mining activity spent on B_M. This design was presumably chosen in [7] because it is really difficult to guarantee and ascertain the correctness of peers local processing, but—as we are going to discuss shortly—it is the root cause of the double-spending threat.
- The "longest chain" is not actually unique. Indeed, it may be the case that the same mining effort corresponds to two or more chains that differs for some block and/or the block order. Thus, also assuming that all the miners follow the majority decision mechanism, branches are possible.
- Each branch must be internally consistent and can never include two conflicting transactions; however, branches do not need to be consistent with one another, and one branch can include a transaction which contradicts a transaction in another branch.
- The blocks that will be incorporated in the blockchain are determined by the upshot of a sequence of mining activities and, ultimately, by the relative fitness of all miners working at that section of the blockchain. As we will detail in Section 4, at least in the case of Bitcoin, this has given rise to a rude mining-hardware race which undermines the confidence of users in the cryptocurrency and its stability on the market.

Branching in a blockchain may result from the fact that different, honest miners are in temporary disagreement about which chain should be considered valid, as they found different "longest chain". That could happen, for example, because of network latencies or network outages.

However, secretly mining a branch is at the core of the double-spending attack. Indeed, in its essence a successful such attack consists in what follows. The attacker P broadcasts to the network a transaction T_P in favor of the seller R. Then P mines a block which builds on the latest block B_l before the one that eventually will contain T_P, and which contains a *conflicting transaction* $\tilde{T}_P$, i.e., a transaction which pays the amount of T_P to another party than R (maybe P herself).

P waits until R is confident in her payment and sends her the merchandise. In the meantime, P uses the secretly mined block to extend the chain in order to get a branch longer than the one known by the network. Finally, she releases the secret branch to the network, thus overriding the payment in favor of R.

A main concept concerning the confidence that a seller can put in a payment in his favor is that of *confirmations of a transaction*. A transaction is said to have c confirmations if it is included in a block which is part of the valid chain, and there are c blocks in the path from that block to the last one of the chain, inclusive [86]. Intuitively, the more confirmations a transaction has the more likely that it will remain in the chain permanently. Nakamoto's paper [7] outlines the number of confirmations versus the relative hashpower of an attacker P, such that the probability of success of P in a double-spending attack is less than 0.1%. From those calculations, it follows that if P has less than 10% of the total hashing power of the network, then $c = 6$ confirmations are sufficient. However, if P has a greater percentage of the hashing power, then an increasingly bigger number c of confirmations is required. For example, if P has 30% of the total hashing power, then $c = 24$ confirmations are required in order to limit the chances of a double-spending attack to less than 1 in 1000. If P has more than 50% of the total hashpower, then sooner or later she will inevitably succeed in a double-spending attack, that is in a "51% attack".

4. Computational Aspects

One important feature in the consensus mechanism implemented in blockchain-based systems is the unpredictability of the miner in charge of recording the next transaction block. First and foremost, this is useful to prevent frauds: with some degree of certainty, and for every single recording, no one should know in advance who would be the peer entitled to record the transaction. If that happened, a malicious user could do a denial-of-service attack against such peer to defraud her of the prize.

At the same time, that unpredictability results in an incentive for peers to do mining, which is vital for a blockchain-based system. Although there are nodes that have some advantage with respect to others in finding the nonce, which ultimately depends on the specific mining approach in use, even the less fitted node has some chance of finding it (see Section 4.1).

In Bitcoin and other blockchain-based systems, another important feature is the time-lapse mechanism implemented through the average time that the miners as a whole spend in mining a new block. Indeed, such average time is somewhat predetermined in each blockchain-based system, and this turns out in a synchronization mechanism for updating the blockchain on every single node in the system.

With this design in mind, and after fixing a specific mining approach, we want to discuss the following two issues: (i) the measure of performance in mining, and (ii) the way the system tunes the difficulty of mining in order to get the preset time-lapse for the blockchain update. In Section 4.1 we will analyze these two concepts to understand how computation is involved in the Bitcoin blockchain implementation.

4.1. Difficulty and Performance in Bitcoin

The average time that the miners as a whole spend in searching the nonce N that solves (2) was predetermined by the Bitcoin creators as about $t = 10$ min. This means that the system should create a new block in the blockchain after spending 10 min on average in searching for a correct digest, and concomitantly establishing (almost randomly) the peer entitled for the block recording.

Since this is a *trial and error* algorithm, the more peers in the system that participate in the race with a fixed τ, the shorter is the average time to find a correct solution. But we know that $t = 10$ min is the predetermined time that the system should take to find a solution, and on average it should be met. This looks like a *soft real-time* system [87], but we think it is not, as the time expected to have a result should not significantly exceed 10 minutes, but it should neither be less than that.

When actual $t < 10$ repeatedly for a number of blocks, then the system increases the difficulty to make the interval longer, that means the threshold τ in (2) must be lower for the next blocks. As we told in Section 3.3.2, lowering the threshold of one single level in Bitcoin corresponds to add one leading zero to the Hex string encoding the previous threshold value or, equivalently, to shift its single "1" digit from position z to $z + 1$. In turn, it can be shown that–at least under the assumption that the hash function is *well balanced* (roughly speaking, in a well balanced hash function each output has about the same number of pre-images [88]) and $z \geq 10$–the above shift results in a proof of work which is probabilistically harder than the previous one by a factor very close to $2^{16} = 65536$.

In Figure 5 we can see how the implemented difficulty increased over the last two years. This is referable to three distinct facts:

- The growth in number of competitors to the race, due to the diffusion of currency and to the prize offered.
- The growth in performance of the single computational resource, due to the usage of better hardware.
- The sentiment of investors in mining pools.

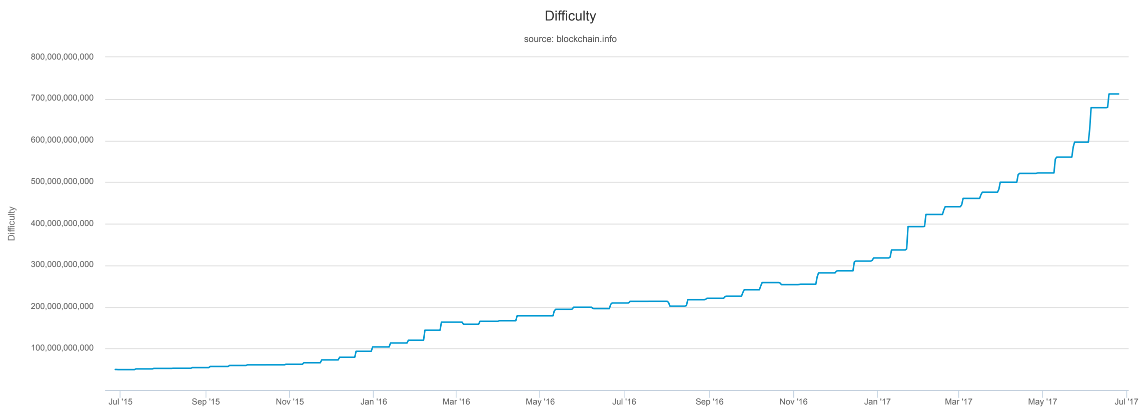

Figure 5. Difficulty of Bitcoin mining in the last two years. It represents the difficulty to find a nonce for a new block when compared to the easiest it can ever be $(easiest\_target/current\_target)$.

The ever growing difficulty, as described in the graph, stagnated starting from March 2016 and even decreased in August 2016. As the news about a better performing mining chip were spreading (AntMiner S9 was released in May 2016), we speculate that miners did not rush in the update of their machines. For more reason, a forthcoming halving of the mining reward was expected by July 2016, and we believe that motivation in energy consuming and further hardware expenses subsequently weakened. But the enthusiasm about this new technology did not diminish, and soon the difficulty started growing again.

But what is a good parameter for performance measuring in this context?

The SHA-256 hash function does not need all the features available in a complex Instruction Set Architecture, as the ones equipped in general purpose computers. Therefore, a new appropriate parameter for evaluating the performance of a hashing system has been introduced: the *hash-rate*, currently measured in TeraHash per second (TH/s). (Notably, two years ago the unit of measure was the GigaHash per second.)

In Figure 6 the global hash-rate of Bitcoin is reported on a daily sample. Since the number of miners is not known, as well as their singular hash-rate, the measurement is calculated from the expected rate of finding a block (144 a day), the actual number of blocks found and the current difficulty, resulting in an oscillating graph. The trending growth is affected also by the improving computing systems adopted.

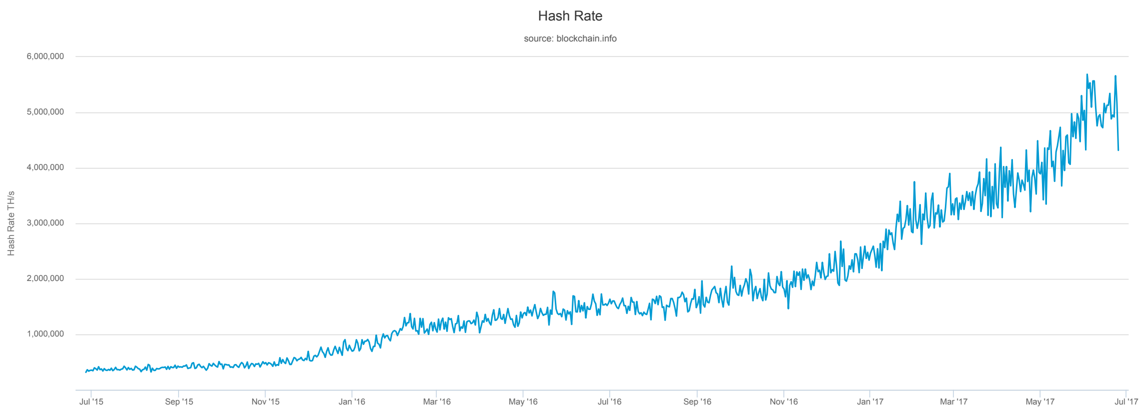

Figure 6. Estimated global hash-rate of the Bitcoin system in the last two years.

But where does this improvement lie in Bitcoin mining systems?

As we described, the solution to our problem can be found through a trial and error algorithm. This immediately leads to the probabilistic observation that the more calculating units the client has, the better chances to win the race.

In fact, in the short lifetime of this cryptocurrency, we have seen a shift in terms of technologies used for the computing resources in the mining competition, from single core to multicores, then GPUs, and FPGAs, to end up with ASICs.

This is the effect of the quest for the correct solution, or-using a nostalgic pleasant term—the Nonce Rush. Anything looking like something useless is sacrificed in the name of the hash-rate. Multicores had a short life since they were inefficient horse carriages. GPUs had a fancy number of lightweight cores working well (with differences due to the hardware implementation of the instructions) but soon they lost appeal. In fact all these technologies seemed like wasted money since someone had the idea to build specific hardware to compute uniquely the double SHA-256 hash function required to check (2). It resulted to be very fast with FPGAs, and incredibly fast with the introduction of ASICs.

As we can see in Table 3, the convenience of using specialized technologies does not lie just in the computing speed, but also in device cost and power consumption. In fact we have to remark again that Bitcoin mining is a computational race based on casualty and attempts. The key to win virtual coins-which are actually usable to buy real goods - is to have the fastest and overall less expensive computing units. Spending thousands dollars without improving the chances to win the race is a terrible investment.

At the same time, if Ms. Smith would decide to make a good investment today by buying the fastest ASIC available, the repayment plan could not be successfully accomplished: in fact a new technology could arise and a certain number of investors could decide to rely on it, making Ms. Smith hardware obsolete and unfit to win any competition. For example, if Ms. Smith bought in 2015 the fastest hardware available, that is the Spondooliestech SP35 Yukon, now she has less than half the probability to find a nonce with respect to an investor that buys the AntMiner S9 today, and she consumes more than the double in power. At the current market price and mining reward, if she finds a nonce with her hardware, she will probably pay back the initial investment, the power consumption and still have a profit. But in the last two years the difficulty grew up for a factor of 14, and her impact in the global Nonce Rush reduced by the same factor due to the growth of the global hash-rate (see Figure 6), so her probability of being rewarded is getting worse every day.

Table 3. Illustrative comparison of different hashing technologies [89,90]. Mhash/s stands for million hashes per second. W is for watt. $ is the estimated cost at the time when the device has been put on the market. Mhash/s/$ is the performance per dollar. Mhash/J is the throughput per watt.

Technology	Device	Mhash/s	W	$	Mhash/s/$	Mhash/J
CPU-Intel	Core i7 3930k	66.6	130	670	0.10	0.51
CPU-AMD	4x Opteron 6174	115	320	220	0.52	0.36
Coprocessor Intel	Xeon Phi 5100	140	225	4000	0.03	0.62
GPU-Nvidia	Tesla S2070	749.23	900	3000	0.25	0.83
GPU-AMD (ATI)	5870x6	2568	1200	300	8.56	2.14
FPGA	Butterflylabs Mini Rig	25,200	1250	15,295	1.64	20.16
ASIC	AntMiner S9	14,000,000	1375	2400	5833	10,182
ASIC	Spondooliestech SP35 Yukon	5,500,000	3650	2235	2460	1506

A common strategy adopted by miners is to team up in mining pools. This significantly raises the probability to find nonces in a big pool, and the rewards are shared among the participants according to the contributed processing power. In this way, risks and profits are more balanced for single miners.

4.2. Making It Useful

When some user talks about a currency, she does not really care about the costs incurred in keeping it working, or better she is interested in her direct costs (interest rates, etc.). At the same time, some supporters of Bitcoin argue that the costs of its current proof-of-work mining system is a fraction of the costs to maintain real material currencies, and it seems a correct claim [91]. But Bitcoin is not as widespread as fiat currencies, and while uncertainty about scaled-up energy costs of mining arises, some alternative initiatives are spreading.

When talking about mining and making computing power available, some user would prefer to give her hardware to some higher purpose than to a consensus mechanism. For that reason, one of the alternative ideas consists in implementing a proof-of-work system while doing some useful calculations.

An example of such a different proof-of-work algorithm is proposed in Primecoin [92], where the computing time is used to find prime number chains known as Cunningham chains and bi-twin chains, both of high scientific interest. This proof-of-work has the property of being efficiently verifiable, through the Fermat test [93] of base 2 together with Euler-Lagrange-Lifchitz test [94], which verify probable primality for the prime chains. Also difficulty can be implemented using the remainder of Fermat test, so that an approximately linear continuous difficulty curve can be constructed for a given prime chain length.

Some developers proposed to use distributed computing tools like folding@home or SETI@home for the currency mining, so that the user computations could be somewhat useful in the cure of protein misfolding diseases or in the search for signals of alien intelligence in the universe.

However the point is that a consensus-based blockchain needs an efficiently verifiable proof-of-work, like SHA-256, but folding@home and SETI@home do not have such a property. Up till now, the participants in the protein folding and SETI networks were volunteers, with probably no intentions other than actually help the goal of the project. If tied to currency mining, these networks could be weakened by participants searching for profits, not bothered with the actual computations.

Instead those miners could be prone to provide fake data that has no value to the humanitarian goals, but that is likely indistinguishable from a genuine output.

Actually there are working implementations of cryptocurrency connected to a protein folding network (CureCoin [95]) and to the BOINC distributed scientific computing network (Gridcoin [96]). In these currencies the research networks are not used for a decentralized consensus, but rather interconnected to another proof-of-work or proof-of-stake system to allot a credit based on the research work done. Let's take CureCoin: in this system the security of the currency is entrusted by nodes running SHA-256 like in Bitcoin, but there is a percentage of the daily mined CureCoins that are granted to the folding@home participants in the system. Indeed those involved in the research job get 76% of the total coins, while blockchain maintainers get another 19%. Finally there is also a sort of development tax: 2% of the total mined coins are distributed to people who donated to project development, while the remaining 3% is reserved to Curecoin developers, and will be used for compensating development costs and for community care. Let us say this would be a fair reward scheme, but it is significantly more centralized than the original Bitcoin one, and probably less resistant to fraud attacks.

A different approach is used in Gridcoin: the consensus is granted with a proof-of-stake, but the amount credited is based on the effective work done in the BOINC network. The interesting point here is that the mining mechanism involves some concepts from Bitcoin, like *network average* which is similar to difficulty in the proof-of-work, but also implements a complex rewarding mechanism which adds layers of interactions between servers and clients. Its finality would be the fairness of rewarding with some degree of fraud resistance, but with several actors involved, the system could overall expose more weaknesses.

Finally, in our view the Nonce Rush implemented in Bitcoin does not have any sense when thinking of a blockchain uncoupled from a currency, as it implies an enormous waste of computing power for mostly useless and costly calculations.

5. Conclusions

Nowadays a new generation of systems relying on concepts and technologies inspired by those of Bitcoin are at the highest international interest because of the prospects both on opportunities and risks. This makes the focus on the features of blockchain-based systems extremely interesting. There are a number of advantages inherently embedded in these systems. First of all they generally exclude any central authority, avoiding in this way a single point of attack. Besides that, they represent an attempt to conjugate both transactor's and recipient's privacy with trusted, publicly verifiable transactions. All this is possible thanks to protocols that combine different cryptographic primitives and schemes into innovative designs, and which in turn pave the way to new very interesting applications with an high impact on society. When talking about currencies, even more advantages emerge. For example merchants are motivated to accept cryptocurrencies because fees are lower than the 2–3% typically imposed by credit card processors, and furthermore, these fees are usually self-imposed and paid by the purchaser, not the vendor. But at the same time some weaknesses are evident.

At present time blockchain-based systems can be seen as "moving targets". Bitcoin incurred in its first hard fork on 1 August , because of the pressure exerted by different groups of people (developers, investors, entrepreneurs, etc.) who need to compete with other systems. This chain split resulted in the newborn Bitcoin Cash (BCC), with a protocol upgrade to fix on-chain capacity. It is unclear which of the two cryptocurrencies will eventually become the de facto Bitcoin. Ethereum—which at the time being has the highest market capitalization after Bitcoin—already incurred in a hard fork, and its developers have formulated an update plan in order to switch from the proof-of-work to a new proof-of-stake called Casper. Moreover, some systems based on promising business models and/or technical innovation (e.g., NEO, IOTA) are quickly gaining market share [9].

Several security threats exist (e.g., 51% attack, other double spending attacks, wallet theft, denial of service attacks) and from time to time some new vulnerability is spotted (see for example the Common Vulnerabilities and Exposures database [97] for Bitcoin), even if the current practice with these system (Bitcoin, first of all) is encouraging and it indicates that there is a successful effort to face attacks or recover from critical mishaps [98].

One of the biggest criticism that we would move against the current design of blockchain-based systems is the demand for anonymity. On one hand, this imposes the introduction of specific costs for managing the blockchain through suitable pricing functions. On the other hand, the proposed anonymity encourages cybercrime but is a weak shield for common people when compared to the power available to governments or criminality. Moreover, the management costs of these systems negatively affects their wide adoption. Indeed the mining problem is actually a kind of artificial puzzle of no practical concern (besides, of course, the existence of the network), which requires an increasing amount of resources (e.g., hashpower, coin age) over the time. Some statistics on Bitcoin report that-at the time of writing-the network hashrate would be equivalent to 78, 918, 182.56 petaflops, which is more than 848, 500 times China's Sunway TaihuLight, the fastest supercomputer in the world [99]. In a time when humanity is facing climate and environment challenges, this is not a feasible option.

Acknowledgments: The authors would like to acknowledge Pasquale Forte for his contribution on a previous version of this work, and the anonymous reviewers for their valuable and constructive suggestions.

Author Contributions: G.S. supervised the paper and worked mainly at the cryptographic and security oriented content, whilst D.R. was expecially focused on the analysis of the computational issues. The two authors worked jointly at describing the current and potential applications of the blockchain technology.

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

References

1. Wikipedia. DigiCash. Available online: https://en.wikipedia.org/wiki/DigiCash (accessed on 13 June 2017).
2. Wikipedia. E-gold. Available online: https://en.wikipedia.org/wiki/E-gold (accessed on 13 June 2017).
3. eDINAR. Available online: https://www.e-dinar.com (accessed on 13 June 2017).
4. Wikipedia. Liberty Reserve. Available online: https://en.wikipedia.org/wiki/Liberty_Reserve (accessed on 13 June 2017).
5. Condon, S. Judge Spares E-Gold Directors Jail Time. Available online: http://www.cnet.com/news/judge-spares-e-gold-directors-jail-time/ (accessed on 28 August 2017).
6. Cloherty, J. 'Black Market Bank' Accused of Laundering $ 6B in Criminal Proceeds. Available online: http://abcnews.go.com/US/black-market-bank-accused-laundering-6b-criminal-proceeds/story?id=19275887 (accessed on 28 August 2017).
7. Nakamoto, S. Bitcoin: A Peer to Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 28 August 2017).
8. Dai, W. B-Money Proposal. 1998. Available online: http://www.weidai.com/bmoney.txt (accessed on 28 August 2017).
9. CryptoCurrency Market Capitalizations. Available online: http://coinmarketcap.com/ (accessed on 11 August 2017).
10. MAPofCOINS. Available online: http://mapofcoins.com/ (accessed on 13 July 2017).
11. Szabo, N. Formalizing and Securing Relationships on Public Networks. 1997. Available online: http://journals.uic.edu/ojs/index.php/fm/article/view/548/469 (accessed on 11 August 2017).
12. Buterin, V. DAOs, DACs, DAs and More: An Incomplete Terminology Guide, 2014. Available online: https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/ (accessed on 20 July 2017).
13. Bitcoin Wiki. Smart Property. Available online: https://en.bitcoin.it/wiki/Smart_Property (accessed on 27 July 2017).
14. Bitcoin Wiki. Colored Coins. Available online: https://en.bitcoin.it/wiki/Colored_Coins (accessed on 27 July 2017).
15. Umeh, J. Blockchain Double Bubble or Double Trouble? *ITNOW* **2016**, *58*, 58–61.
16. Proof of Existence. Available online: http://www.proofofexistence.com/ (accessed on 13 June 2017).
17. Everledger. Available online: https://www.everledger.io/ (accessed on 18 July 2017).
18. Official Kimberley Process. Available online: https://www.kimberleyprocess.com (accessed on 18 July 2017).

19. Wright, A.; De Filippi, P. Decentralized Blockchain Technology and the Rise of Lex Cryptographia, 2015. Available online: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2580664 (accessed on 3 July 2017).
20. Swan, M. *Blockchain: Blueprint for a New Economy*; O'Reilly Media, Inc.: California, CA, USA, 2015.
21. Delmolino, K.; Arnett, M.; Kosba, A. A Programmer's Guide to Ethereum and Serpent, 2015. Available online: https://mc2-umd.github.io/ethereumlab/docs/serpent_tutorial.pdf (accessed on 28 August 2017).
22. Solidity Documentation. Available online: https://solidity.readthedocs.io/en/develop/ (accessed on 8 June 2017).
23. ICANN (Internet Corporation for Assigned Names and Numbers). Available online: https://www.icann.org/ (accessed on 14 July 2017).
24. Ramaswamy, V.S.; Namakumari, S. *Marketing Management: Global Perspective, Indian Context*; McGraw Hill Education (India) P. Ltd.: New Delhi, India, 2013.
25. Namecoin. Available online: https://namecoin.org/ (accessed on 14 June 2017).
26. Buterin, V. Ethereum White Paper, 2014. Available online: https://github.com/ethereum/wiki/wiki/White-Paper#ethereum (accessed on 3 July 2017).
27. Frontier Documentation. Available online: https://ethereum.gitbooks.io/frontier-guide/content/index.html (accessed on 20 July 2017).
28. A Call for a Temporary Moratorium on "The DAO". Available online: https://docs.google.com/document/d/10kTyCmGPhvZy94F7VWyS-dQ4lsBacR2dUgGTtV98C40 (accessed on 20 July 2017).
29. The DAO, The Hack, The Soft Fork and The Hard Fork. Available online: https://www.cryptocompare.com/coins/guides/the-dao-the-hack-the-soft-fork-and-the-hard-fork/ (accessed on 20 July 2017).
30. The Explorer for the Ethereum blockchain. Available online: https://etherchain.org/ (accessed on 20 July 2017).
31. The Ethereum Block Explorer. Available online: https://etherscan.io/ (accessed on 20 July 2017).
32. Ethereum Classic Block Explorer. Available online: http://gastraker.io (accessed on 20 July 2017).
33. ETC E CHAIN. Available online: https://etcchain.com/explorer (accessed on 20 July 2017).
34. Enterprise Ethereum Alliance. Available online: https://entethalliance.org/ (accessed on 20 July 2017).
35. Storj Whitepaper, 2016. Available online: https://storj.io/storj.pdf (accessed on 28 August 2017).
36. Pureswaran, V.; Brody, P. Device democracy, Saving the future of the Internet of Things, 2015. Available online: http://www-935.ibm.com/services/multimedia/GBE03620USEN.pdf (accessed on 28 August 2017).
37. Telehash. Available online: http://telehash.org/ (accessed on 20 July 2017).
38. BitTorrent. Available online: http://www.bittorrent.com/ (accessed on 27 July 2017).
39. Morabito, V. *Business Innovation Through Blockchain: The B3 Perspective*; Springer International Publishing: Berlin/Heidelberg, Germany, 2017.
40. Carlisle, B.G. Proof of Prespecified Endpoints in Medical Research with the Bitcoin Blockchain. Available online: https://www.bgcarlisle.com/blog/2014/08/25/proof-of-prespecified-endpoints-in-medical-research-with-the-bitcoin-blockchain/ (accessed on 20 July 2017).
41. Nichol, P.B. Blockchain Applications for Healthcare. Available online: http://www.cio.com/article/3042603/innovation/blockchain-applications-for-healthcare.html (accessed on 20 July 2017).
42. Walch, A. The bitcoin blockchain as financial market infrastructure: A consideration of operational risk. *N. Y. Univ. J. Legis. Public Policy* **2015**, *18*, 837.
43. Tapscott, A.; Tapscott, D. How Will Blockchain Change Banking? How Won't It? Available online: https://www.americanbanker.com/opinion/how-will-blockchain-change-banking-how-i-wont-i-it (accessed on 24 July 2017).
44. Brennan, C.; Lunn, W. *Blockchain: The Trust Disrupter*; Credit Suisse Securities (Europe) Ltd.: London, UK, 2016.
45. Ripple. Available online: https://ripple.com/ (accessed on 24 July 2017).
46. Crosman, P. Ripple vs. Swift Rivalry Heats Up; Banking May Be Ultimate Winner. Available online: https://www.paymentssource.com/news/ripple-vs-swift-rivalry-heats-up-banking-may-be-ultimate-winner (accessed on 24 July 2017).
47. Tschorsch, F.; Scheuermann, B. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2084–2123.
48. Gutierrez, C.M.; Gallagher, P. (Eds.) *Secure Hash Standard*; Federal Information Processing Standards Publication; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2008; Volumes 180–183.

49. Pritzker, P.; May, W. (Eds.) *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*; Federal Information Processing Standards Publication; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2015; Volume 202.
50. Bitcoin.info Explorer. Available online: https://blockchain.info (accessed on 23 July 2017).
51. Bitcoin Block Explorer. Available online: https://blockexplorer.com/ (accessed on 13 July 2017).
52. Bitcoin Network Graphs by bitcoin.org. Available online: http://bitcoin.sipa.be/ (accessed on 13 July 2017).
53. Bitcoin Wiki.Script. Available online: https://en.bitcoin.it/wiki/Script (accessed on 21 July 2017).
54. Cooper, D.; Santesson, S.; Farrell, S.; Boeyen, S.; Housley, R.; Polk, W. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. 2008. Available online: http://tools.ietf.org/html/rfc5280 (accessed on 28 August 2017).
55. Johnson, D.; Menezes, A.; Vanstone, S. The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63.
56. Dobbertin, H.; Bosselaers, A.; Preneel, B. RIPEMD-160: A strengthened version of RIPEMD. In *Fast Software Encryption*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 71–82.
57. Bitcoin Developer Guide. Available online: https://bitcoin.org/en/developer-guide (accessed on 14 July 2017).
58. Menezes, A.J.; Van Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 2010.
59. Bos, J.W.; Halderman, J.A.; Heninger, N.; Moore, J.; Naehrig, M.; Wustrow, E. Elliptic curve cryptography in practice. In Proceedings of the International Conference on Financial Cryptography and Data Security, Christ Church, Barbados, 3–7 March 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 157–175.
60. Ron, D.; Shamir, A. Quantitative analysis of the full bitcoin transaction graph. In Proceedings of the International Conference on Financial Cryptography and Data Security, Okinawa, Japan, 1–5 April 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 6–24.
61. Fleder, M.; Kester, M.S.; Pillai, S. Bitcoin transaction graph analysis. *arXiv* **2015**, arXiv:1502.01657.
62. Garay, J.; Kiayias, A.; Leonardos, N. The bitcoin backbone protocol: Analysis and applications. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, 26–30 April 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 281–310.
63. Tor project. Overview. Available online: https://www.torproject.org/about/overview.html.en (accessed on 14 July 2017).
64. Wikipedia. Forth Programming Language. Available online: https://en.wikipedia.org/wiki/Forth_(programming_language) (accessed on 17 July 2017).
65. Bitcoin Wiki. Contract. Available online: https://en.bitcoin.it/wiki/Contract (accessed on 21 July 2017).
66. Wood, G. Ethereum: A Secure Decentralized Generalized Transaction Ledger, 2016. Available online: https://ethereum.github.io/yellowpaper/paper.pdf (accessed on 28 August 2017).
67. Bitcoin Wiki. Block Size Limit Controversy. Available online: https://en.bitcoin.it/wiki/Block_size_limit_controversy (accessed on 21 July 2017).
68. StephenM347. Bitcoin@Stack Exchange. What's the Purpose of a Maximum Block Size? Available online: http://bitcoin.stackexchange.com/questions/37292/whats-the-purpose-of-a-maximum-block-size/37303#37303 (accessed on 14 July 2017).
69. Merkle, R.C. A digital signature based on a conventional encryption function. In *Advances in Cryptology—CRYPTO'87*; Springer: Berlin/Heidelberg, Germany, 1988; pp. 369–378.
70. Bitcoin Wiki. Block Hashing Algorithm. Available online: https://en.bitcoin.it/wiki/Block_hashing_algorithm (accessed on 27 July 2017).
71. Shirrif, K. Bitcoin Mining the Hard Way: The Algorithms, Protocols, and Bytes. Available online: http://www.righto.com/2014/02/bitcoin-mining-hard-way-algorithms.html (accessed on 11 August 2018).
72. Jakobsson, M.; Juels, A. Proofs of Work and Bread Pudding Protocols. In Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security, Deventer, The Netherlands, 20–21 September 1999; pp. 258–272.
73. Dwork, C.; Naor, M. Pricing via processing or combatting junk mail. In *Advances in Cryptology—CRYPTO'92*; Springer: Berlin/Heidelberg, Germany, 1993; pp. 139–147.
74. Rogaway, P. Nonce-Based Symmetric Encryption. In *Fast Software Encryption*; Roy, B., Meier, W., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3017, pp. 348–358.

75. Rosenfeld, M. Analysis of Hashrate-Based Double Spending. 2014. Available online: http://arxiv.org/abs/1402.2009v1 (accessed on 28 August 2017).
76. Litecoin. Available online: https://litecoin.org/ (accessed on 20 July 2017).
77. Percival, C. Stronger Key Derivation via Sequential Memory-Hard Functions, 2009. Available online: http://www.bsdcan.org/2009/schedule/attachments/87_scrypt.pdf (accessed on 28 August 2017).
78. Rivest, R. The MD5 Message-Digest Algorithm. 1992. Available online: http://tools.ietf.org/html/rfc1321 (accessed on 28 August 2017).
79. Daley, W.M.; Kammer, R.G. (Eds.) *Data Encryption Standard (DES)*; Federal Information Processing Standards Publication; National Institute of Standards and Technology: Gaithersburg, MD, USA, 1999; Volume 46–3.
80. King, S.; Nadal, S. Ppcoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake, 2012. Available online: https://peercoin.net/assets/paper/peercoin-paper.pdf (accessed on 28 August 2017).
81. Larimer, D. Transactions as Proof-of-Stake, 2013. Available online: https://steemit.com/bitshares/@testz/bitshares-history-transactions-as-proof-of-stake-tapos (accessed on 3 July 2017).
82. Larimer, D. Delegated Proof-of-Stake (DPoS), 2014. Available online: http://docs.bitshares.eu/bitshares/dpos.html (accessed on 28 August 2017).
83. Takura, A.; Ono, S.; Naito, S. A secure and trusted time stamping authority. In Proceedings of the IEEE IWS 99 Internet Workshop, Osaka, Japan, 18–20 February 1999; pp. 88–93.
84. Bentov, I.; Lee, C.; Mizrahi, A.; Rosenfeld, M. Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake. *SIGMETRICS Perform. Eval. Rev.* **2014**, 42, 34–37.
85. Wuille, P. Bitcoin@Stack Exchange. What Does the Term "Longest Chain" Mean? Available online: http://bitcoin.stackexchange.com/questions/5540/what-does-the-term-longest-chain-mean/5542#5542 (accessed on 14 July 2017).
86. Kolden, J. Bitcoin@Stack Exchange. What Are Bitcoin "Confirmations"? Available online: http://bitcoin.stackexchange.com/questions/146/what-are-bitcoin-confirmations/160#160 (accessed on 14 July 2017).
87. Kopetz, H. *Real-Time Systems: Design Principles for Distributed Embedded Applications*; Springer: Berlin/Heidelberg, Germany, 2011.
88. Laccetti, G.; Schmid, G. Brute force attacks on hash functions. *J. Discret. Math. Sci. Cryptogr.* **2007**, *10*, 439–460.
89. Bitcoin Wiki. Non-Specialized Hardware Comparison. Available online: https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison (accessed on 15 June 2017).
90. Bitcoin Wiki: Mining Hardware Comparison. Available online: https://en.bitcoin.it/wiki/Mining_hardware_comparison (accessed on 15 June 2017).
91. McCook, H. An Order-of-Magnitude Estimate of the Relative Sustainability of the Bitcoin Network, 2015. Available online: https://www.academia.edu/7666373 (accessed on 28 August 2017).
92. King, S. Primecoin: Cryptocurrency with Prime Number Proof-of-Work, 2013. Available online: http://primecoin.io/bin/primecoin-paper.pdf (accessed on 28 August 2017).
93. The University of Tennessee. Finding Primes and Proving Primality. Available online: http://primes.utm.edu/prove/merged.html (accessed on 10 August 2017).
94. Lifchitz, H. Generalization of Euler-Lagrange Theorem. Available online: http://www.primenumbers.net/Henri/us/NouvTh1us.htm (accessed on 10 August 2017).
95. CureCoin. What Is CureCoin? Available online: https://curecoin.net/knowledge-base/about-curecoin/what-is-curecoin/ (accessed on 27 July 2017).
96. Gridcoin Wiki. Available online: http://wiki.gridcoin.us/Main_Page (accessed on 27 July 2017).
97. Common Vulnerabilities and Exposuresdatabase. Available online: https://cve.mitre.org/cve/cve.html (accessed on 27 July 2017).
98. Wikipedia. History of Bitcoin. Available online: https://en.wikipedia.org/wiki/History_of_Bitcoin (accessed on 14 July 2017).
99. Top 500 Supercomputing Sites. Available online: www.top500.org (accessed on 30 July 2017).