Received 28 August 2017
Revised 25 September 2017
Accepted 29 September 2017

Blockchain: properties and misconceptions

Daniel Conte de Leon, Antonius Q. Stalick and Ananth A. Jillepalli
Center for Secure and Dependable Systems and Computer Science Department, University of Idaho, Moscow, Idaho, USA

Michael A. Haney
Center for Secure and Dependable Systems and Center for Advanced Energy Studies and Computer Science Department, University of Idaho, Idaho Falls, Idaho, USA, and

Frederick T. Sheldon
Computer Science Department, University of Idaho, Moscow, Idaho, USA

Abstract

Purpose – The purpose of this article is to clarify current and widespread misconceptions about the properties of blockchain technologies and to describe challenges and avenues for correct and trustworthy design and implementation of distributed ledger system (DLS) or Technology (DLT).

Design/methodology/approach – The authors contrast the properties of a blockchain with desired, however emergent, properties of a DLS, which is a complex and distributed system. They point out and justify, with facts and analysis, current misconceptions about the blockchain and DLSs. They describe challenges that these systems will need to address and possible solution avenues for achieving trustworthiness.

Findings – Many of the statements that have appeared on the internet, news and academic articles, such as immutable ledger and exact copies, may be misleading. These are desired emergent properties of a complex system, not assured properties. It is well-known within the distributed systems and critical software community that it is extremely hard to prove that a complex system correctly and completely implements emergent properties. Further research and development for trustworthy DLS design and implementation is needed, both practical and theoretical.

Research limitations/implications – This is the first known published attempt at describing current misconceptions about blockchain technologies. Further collaborative work, discussions, potential solutions, evaluations, resulting publications and verified reference implementations are needed to ensure DLTs are safe, secure, and trustworthy.

Practical implications – Interdisciplinary teams with members from academia, business and industry, and from disciplines such as business, entrepreneurship, theoretical and practical computer science,

Asia Pacific Journal of Innovation and Entrepreneurship
Vol. 11 No. 3, 2017
pp. 286-300
Emerald Publishing Limited
2071-1395
DOI 10.1108/APJIE-12-2017-034

The authors would like to thank the US National Science Foundation (NSF), the State of Idaho and the University of Idaho for funding this research work: NSF under CyberCorps(R) awards 1027409 and 1565572; and the State of Idaho under the IGEM Cybersecurity Management for Cyber-Physical Systems capacity award. These funding entities were not directly involved in the preparation of this article. The opinions expressed in this article are not necessarily those of the US Government or the State of Idaho. This paper forms part of a special section on 'Blockchain on business and entrepreneurship'.

cybersecurity, finance, mathematics and statistics, must be formed. Such teams must collaborate with the objective of developing strategies and techniques for ensuring the correctness and security of future DLSs in which our society may become dependent.

Originality value – The value and originality of this article is twofold: the disproving, through fact collection and systematic analysis, of current misconceptions about the properties of the blockchain and DLSs, and the discussion of challenges to achieving adequate trustworthiness along with the proposal of general avenues for possible solutions.

Keywords Blockchain, Distributed agreement, Distributed ledger systems, Technologies and protocols, Emergent system properties, Secure and correct systems, Smart contracts

Paper type Research paper

1. Introduction and overview

In a distributed model, storage and computation are shared between member users, or nodes, connected to a peer-to-peer network. It has been proposed that distributed models, based on blockchain-based ledgers, also called distributed ledger technology (DLT) or system (DLS), could offer higher service availability at much lower costs for some types of business and enterprise applications. For example, property ownership and transfer ledgers, financial trading ledgers, digital currency ledgers and worldwide food traceability tracking applications, among many others. DLT, under the right conditions, may offer increased service availability and resiliency for many digital services afforded by distributed storage, compute, and control.

This distributed nature offers some advantages and creates many opportunities. However, this distributed nature also creates major challenges and major research questions. These questions need to be answered before DLTs demonstrate high levels of security and trust. It is important to carefully investigate the advantages, disadvantages and risks of a DLS, and that the system is implemented correctly before it is used by the public, commerce, and industry. It is also important that DLS designers, implementers, and users understand the advantages and limitations of the technology behind these systems.

In this article, we first describe business and application areas where blockchain technology has been stated capable of upending the current status quo (Section 2). We next introduce blockchains and describe and analyze current misconceptions about their properties (Section 3). We then describe current challenges for improving the trustworthiness of blockchain designs and implementations (Section 4). Next, we describe emergent, and currently unproven, properties of DLSs stressing the distinction between intrinsic blockchain properties and emergent system properties (Section 5). Following, we describe some of the challenges that DLSs, as applied to digital and distributed financial and contractual transactions, will need to address before they are demonstrated trustworthy (Sections 6 and 7). Lastly, we describe how DLSs that combine well-studied secure and trustworthy designs with symmetric and asymmetric encryption, an adequate public key infrastructure and certification authorities, and encrypted computation techniques with correct implementations could begin to lead widespread adoption and the creation of successful new distributed application markets (Section 8). Finally, we present our conclusions in Section 9 and follow with acknowledgements and a complete list of references.

2. Applications and motivation

Most current cloud-based application services rely on a single trusted controlling organization that manages the storage, compute and networking (Infrastructure as a

Service: IaaS), and applications (Software as a Service: SaaS). Access control to associated services and data is also determined by the organization. Such approaches may be desirable for some applications and sometimes necessary given the business and regulatory environment. However, in some cases, a decentralized and distributed approach may be better suited for the business case. Because of this, it is possible that DLT could dramatically change several applications in the business-to-business world. For example, Walmart, the world's largest retailer, is conducting pilot tests for worldwide supply chain food traceability (Popper and Lohrmarch, 2017), which is essential for food safety. Santander Bank, the largest bank in the Eurozone with 13,000+ branches worldwide, has been piloting a remote currency transfer app (Santander, 2015). These pilot applications are based on DLT, which may offer a more efficient system for these types of applications.

Goldman and Sachs (Schneider *et al.*, 2016) has identified five areas where they believe that DLT has potential for major disruption: sharing economy, distributed electricity generation and markets, property ledgers, security markets and financial transactions. Many major financial institutions have started to experiment and pilot the technology as a means for keeping a distributed verifiable and integral financial ledger (Norton, 2016). IBM has invested a considerable amount of resources in developing the technologies for these protocols to be ready for corporate use and has developed the Hyperledger software platform which is an enterprise-quality implementation of DLT. Also, in 2016, IBM led the creation of the Hyperledger Foundation, which now has 130 corporate members and leads the Hyperledger open-source software platform (Hyperledger, 2017).

It is of economic value that we pursue and develop new technologies for business and commerce. In addition, it is essential that we perform the research and analysis needed to ensure these new technologies are designed and implemented for correctness, security, privacy and trustworthiness. There are countless examples of technologies and products being adopted before being demonstrated safe, secure and trustworthy, which have sadly resulted in immeasurable economic losses and loss of life.

Most of the properties of the blockchain that have been stated lately as intrinsic such as *immutable* and *exact copy*, among others, are not intrinsic properties of blockchains but desired and emergent properties of a complex system with many users or agents, some or all of which may not be trusted. It is very well-known in the distributed systems and software engineering community that proving emergent properties of a complex distributed system is an extremely hard endeavor.

In this paper, we present a detailed analysis of the properties of blockchains and DLT and the challenges that these technologies still face to be ready for mainstream business adoption without increasing the risk to businesses and society's well-being due to design or implementation vulnerabilities and potential cyberattacks that may exploit those vulnerabilities.

3. Properties and misconceptions about blockchains

In this section, we introduce blockchains, describe their properties and analyze current misconceptions.

3.1 What is a blockchain?

A blockchain is a digital information recording method capable of recording data using a logbook approach and with the following essential characteristics: 1-Ordered, 2-Incremental, 3-Sound (cryptographically verifiable upto a given block) and 4-Digital. Other characteristics such as distributed and mutable-by-proof-of-work are not characteristics of a blockchain itself but characteristics added by sharing, distribution, communication, and

agreement protocols. A blockchain is composed of a cryptographically linked chain of blocks of data. Blocks are chained in a sequence using cryptographic hashes. A hash is a fixed length number derived from a given message or document. Each block is composed of three major components:

(1) *Block-Data*: a set of messages or transactions;

(2) *Chaining-Hash*: a copy of the hash value of the immediately preceding block; and

(3) *Block-Hash*: the calculated hash value of the data block or messages plus the chaining hash value in 2 (Norton, 2016, Gupta, 2017, Nakamoto, 2008).

For additional details on cryptographic hashes, we refer the reader to Stallings and Brown (2015) and NIST (2015a, 2015b). In Figure 1, we present a simplified graphical representation of the process for requesting and storing transactions within a DLS that uses a blockchain as its distributed storage medium.

3.2. Properties of a blockchain

The immutability property is an emergent property of a DLS and not an intrinsic property of a blockchain data structure. It is well-known within the software engineering community that verifying emergent system properties in complex systems is an extremely hard problem. Here, we clarify what can be logically inferred and what cannot from the by-construction characteristics of a free-standing blockchain. Later in this article we analyze the characteristics of a DLS.

Currently, many websites, books, professional reports and news and academic articles state that blockchains are *immutable* or *unchangeable*, and that *transactions in a blockchain*

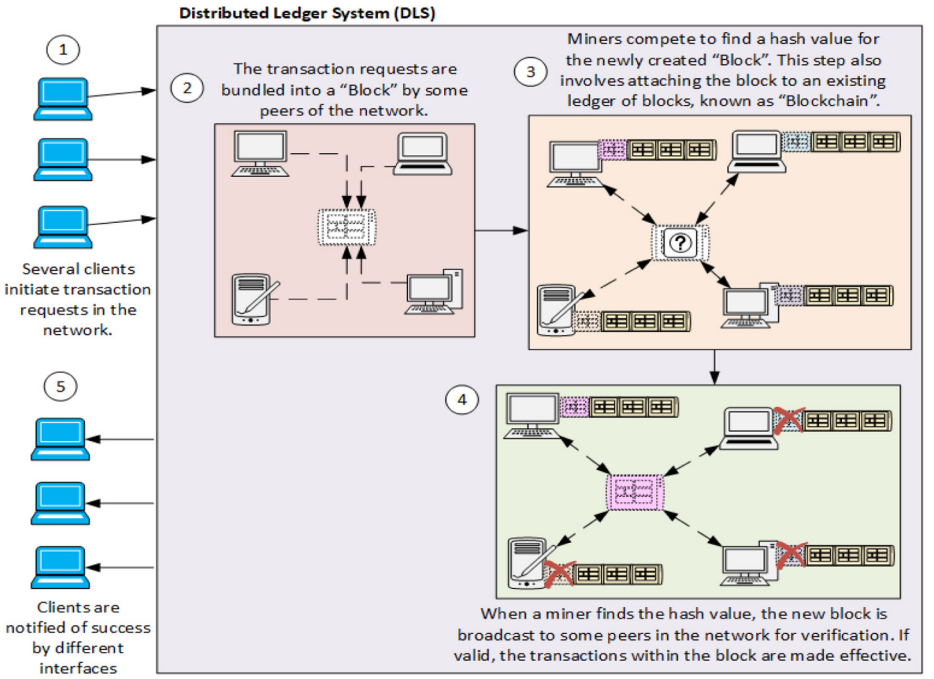

Figure 1. Simplified graphical view of a distributed ledger system

cannot be modified. As written, these statements are incorrect and misleading. The analysis presented in this article plus recent events such as the Ethereum Decentralized Autonomous Organization (DAO) fork (Buterin, 2016b) described later in this article clearly disprove such immutability claims. In the case of a nonced proof-of-work (PoW) distributed blockchain, computational work is needed to modify its data while preserving the soundness, up to the strength of the hash function used. This does not mean that such a blockchain is immutable, but that an agent or set of agents with a sufficient amount of computing power has modified it, perhaps collaboratively. In this case, a more adequate term would be: Mutable-By-Hashing-Power.

It is true with cryptographic certainty, due to the properties of cryptographic hashes, that a positive detection of tampering can be detected for a given blockchain by the failure to verify its soundness. This is true only under the assumptions that the given blockchain was sound (well-formed) when created, the detection is with a high degree of cryptographic certainty but not absoluteness, and subject to the current strength of the hash function used.

However, when observing a freestanding blockchain, if the blockchain is sound, then the only valid conclusion we can make is that the blockchain is well-formed. We cannot conclude that the blockchain has not been modified or that it has not been tampered with. In other words, if the soundness check fails, then we can conclude that *something was modified*. If the soundness property holds, we cannot conclude that *nothing was modified*. The *tamper-evident* term is being used in the literature as implying the latter. However, the former is tampering evidence by the lack of soundness. The latter requires full provenance accountability in addition to the former. To be able to conclude the latter, we must prove an emergent property of a DLS, and not just a property of the underlying data structure or blockchain such as soundness.

If the blockchain is a nonced blockchain and by the soundness requirement all block hash values comply with a specified criterion, usually of being close to zero by an arbitrary delta, then an agent or set of agents that have modified or tampered with the blockchain must have spent a given amount of computational power for each block that was modified. This computational power is needed to search for new block nonce values and recalculate the corresponding new block hash values such that the soundness of the blockchain is rebuilt after modification. Due to the ordered property of a blockchain, this must be done in sequence, starting with the block that was modified and following with all blocks forward in the chain.

The property of ledger *immutability* is a desired but emergent property of a DLS. It may be that in the future it could be mathematically or statistically demonstrated that for a given system a certain set of emergent properties hold, at a given point in time, or under certain conditions. A few researchers have begun to formally and experimentally analyze these properties on current DLS designs and implementations (Gervais *et al.*, 2016; Wüst and Gervais, 2016). However, more research on formal models and analyses, formal and stochastic, and theoretical and practical, need to be performed to demonstrate with measured certainty under which conditions and implementation parameters these emergent properties would hold for a given DLS.

4. Challenges on blockchains

We believe there are three major challenges for the design and implementation of industry-ready blockchains:

(1) the selection of time-resilient strong cryptographic hash functions or the design of hash function vulnerability mitigation techniques;

(2) the correct and secure design and implementation of code that implements a DLS; and

(3) the adequate selection and implementation of efficient, robust and trustworthy leader election algorithms.

We elaborate on these below.

Cryptographic hash functions and asymmetric cryptography are based on what is called *one-way* functions. Some mathematical problems and their corresponding calculations are computationally easy in one direction but computationally infeasible in the inverse direction. As a simple example, given the two numbers 11 and 17, it is computationally easy to calculate their product $11 \times 17 = 187$. However, it is computationally harder to infer that 187 resulted from 11×17. A composite number could have resulted from different combinations of numbers being multiplied. Secure hashes in the secure hash algorithm (SHA)-2 specification use numbers of 224 to 512 binary digits in length. Bitcoin and Ethereum use SHA-256.

One challenge for trustworthy blockchain design is that the property of *computationally infeasible* is not a static property of a given hash function but it changes with the advancement of mathematics and computing. A hash algorithm that is cryptographically secure (computationally infeasible to find the inverse) today may not be cryptographically secure 20 years from now. This hampers our ability to design DLS that can withstand the test of time. Such time resiliency would be required for property ownership ledgers, among others. We believe that it would be important to design and evaluate DLSs that can seamlessly accommodate changes of the hash function. For example, a nested blockchain approach could achieve this by wrapping the previous generation blockchain, or portions of it, on one or more blocks of the new generation. Research and development in this area is much needed.

The issue of correct and secure implementation affects the blockchain data structure as well as the complete DLS. If DLSs are to be in widespread use in business and commerce, it is essential that the design and code that implements these systems has been developed with the highest security and correctness assurances. This today means the use of adequate software engineering methods for design and development, the use of formal methods for protocol specification and property verification, and thorough and exhaustive testing. This is in contrast to ad-hoc approaches that may exhibit much higher probabilities of resulting in incorrect and insecure software designs, implementations, deployments and configurations.

5. Current misconceptions about distributed ledger systems

In this section, we analyze and justify current misconceptions about properties of DLSs.

5.1. What is a distributed ledger system

A DLS, based on DLT, is a computer-based system in which a set of computer processes representing agents or users connected to a digital network operate collaboratively on a set of distributed ledger data structures. The prime data structure here is a blockchain, as defined and analyzed in the two previous sections.

5.2. Emergent properties of a DLS

First, we define *emergent ledger* or *emergent blockchain*. An emergent ledger or blockchain, in a DLS, is the resulting ledger for which the *majority* of the users on the network agree at

any given time. Notice that *majority* here is defined by DLS's protocols. Ideally, a system should converge to a unique *emergent ledger* as fast as possible. However, speed of convergence, size of consensus, and the resulting emergent ledger are all *emergent* properties of a DLS, not by construction or automatically assured properties.

An emergent ledger or *emergent blockchain* is the data backbone of a DLS. It is a virtual entity that exists only as an emergent property of a DLS. It can be modified by agreement of some of the agents in the network. It is important not to confuse a given blockchain data structure with an *emergent blockchain*. The latter is an emergent property of a complex system. Trust in such emergent blockchain is the result of trust in the complete system, which is composed of many components and interacting processes, protocols, and agents. Agents, which may operate within or outside the expected rules and protocols, either by intent or by error, malicious or benign. A DLS should present the following characteristics:

- *Convergent Ledger*: The protocols in a DLS must demonstrate, with a high level of assurance, that the distributed ledger or blockchain converges to a unique emergent blockchain. Many current websites and descriptions of DLT state that every user has, by design, an identical copy of a unique ledger or blockchain. This statement is incorrect and misleading. It is well-known that in an Asynchronous Distributed System, every agent has *a copy* of the data. This copy may or may not be the same copy for all agents. A *goal*, of a DLS, is for all copies of the ledger or blockchain to converge to the same value or state. However, this is a goal and not an intrinsic property.
- *Timely Convergence*: The protocols in a DLS must demonstrate, with a high level of assurance, that the emergent ledger or blockchain will converge within a bounded amount of time.

Difficulty in Proving These: Both properties stated above are emergent and very difficult to prove for a given system. In fact, it is not possible to prove both at the same time in the general case of asynchronous distributed systems with failures. The Fischer *et al.* (1983) article, very well-known within the distributed systems community, states that for the case of arbitrary asynchronous distributed systems, a deterministic consensus protocol does not exist, even with just one fail-stop process (Fischer *et al.*, 1983). This is known in the literature as the Fischer, Lynch, and Patterson result.

5.3. Current misconceptions about DLS

In a DLS, there is not a single unique global blockchain as it has been stated or implied by several sources in the literature and the internet; rather, there is the expectation of an emergent and converging blockchain. In a DLS, each device on a peer-to-peer network keeps its own "copy" of the blockchain. The system has two functional goals:

(1) enable agents to request new transactions on the distributed ledger; and

(2) enable and promote agreement or consensus on a common ledger (the emergent blockchain).

It is important to note that these goals are not intrinsic properties of a blockchain or DLS. These properties are emergent properties of the system. It is possible that in a DLS that has been designed, implemented or configured incorrectly, that one or both goals are not assured.

In some current descriptions in the literature, it is incorrectly assumed that all users or agents of the same type are running the same trusted or original software, using the same

data models, and hence implementing the same protocols. There is no way to ensure such assumptions. There is also no practical reason for doing so. From the point of view of the DLS, it does not matter which software generated the messages on the network. What matters to the DLS is that the messages are well-formed and authenticated. All agents must assume that the software running on peer agents may not be the same and may not implement the same protocol or behave as expected. A DLS must be able to achieve its goals without making any assumptions about the software that generated the messages going through its network. Messages are either well-formed or they should be discarded. If messages must also be authenticated, then they must be signed with a valid private key or certificate that has not been revoked.

Also, depending on the protocols and mining strategy, agents could assume a role as needed instead of being of a different type. This is currently not practical in proof-of-work systems. The hash throughput of a workstation cannot be fairly compared with that of a mobile device. In the case of Bitcoin and Ethereum, which are based on SHA-256 hashes, Application-Specific Integrated Circuits (ASICs) are currently used for fast hash throughput for miner agents. These are hardware devices created specifically to perform SHA-256 calculations and are measured in Million Hashes Per Second. A team of general-purpose central processing unit (CPUs), or a team of graphics processing units, cannot compete with a set of SHA-256 ASICs. By contrast, proof-of-elapsed-time approaches depend on dedicated trusted hardware components within new Intel CPUs (Intel, 2017) rather than on hashing throughput.

It has also been stated in the literature that a lottery style protocol for leader election is used in proof-of-work systems such as Bitcoin and Ethereum. This under the premise that the winner of the lottery for a given block is the agent that has found a nonce that makes the block hash value smaller than the current protocol configured value. However, there are many differences between a lottery style drawing and a proof-of-work system. In a proof-of-work system, agents with higher hash compute power have higher chances at finding a matching hash value. In addition, the cost of a hash per second is not the same for all agents. Furthermore, equalizing the hash-per-second cost requires high up-front investment from the agents. In a lottery, all equivalent entries cost the same amount, it is impossible for any single agent to acquire a fairness-threatening amount of entries, and the reward and the cost of the entries are both monetary. In the case of proof of work by mining DLSs, none of these properties hold.

In addition, other approaches to leader election have been proposed for DLSs. The survivability and distributed systems community have been investigating issues of leader election and distributed agreement for many years. Several innovative and potential solutions have resulted from years of research and development in this area. This area of improvement is perhaps the most urgent area for DLSs. This is because in a proof-of-work by hash mining DLS, such as Bitcoin and Ethereum, the monetary cost and power consumption of mining can grow very rapidly, resulting in an unsustainable and economically unviable system. Innovative robust and efficient approaches to distributed leader election must be developed, implemented and tested within the context of DLTs. It is imperative that results from the survivability and distributed systems research community be considered and applied within this new context.

6. Challenges with cryptocurrencies and Bitcoin

Centralized solutions that provide remote currency transfer services rely on pre-established trust between the central authority and the parties in the transaction. The central authority, which in most countries is a financial entity regulated by law, ensures user authentication

and mediates all transactions. Financial transactions must be authentic, one-time, verifiable and irreversible unless proven fraudulent. The trusted central authority ensures these properties. The trusted authority is also in charge of establishing rules and processes for preventing fraud. That is, preventing malicious users from attempting financial cyberattacks such as spending currency they do not have or multiple times, denying made transactions, creating incomplete transactions or impersonating other users or agents.

Distributed cryptocurrencies do not rely on a central trusted authority; rather the integrity of the system is an emergent property of the DLS. Cryptocurrency systems enable users to transfer cryptocurrency remotely between two parties that have joined the cryptocurrency network across the internet and without the use of a designated and trusted mediator. The motivating application for DLT was the Bitcoin cryptocurrency (Nakamoto, 2008). Currently, all other cryptocurrencies are also based on DLT, albeit with some differences with the originating Bitcoin protocol. In these cases, the mediation and approval of the transaction is made by agreement within the network and as specified by the consensus protocol. The transaction is stored on an emergent blockchain. The blockchain is not unique, contrary to what is stated by many other sources, and not all users may observe the same blockchain at the same time. This is a problem intrinsic to all distributed asynchronous systems.

In *Bitcoin: A Peer-to-Peer Electronic Cash System* (Nakamoto, 2008), the creator(s) of the Bitcoin protocol described what is known as the first cryptocurrency system. The Bitcoin system uses a blockchain as a transaction ledger. The value of a given wallet, or Bitcoin address, is calculated by means of the result of all transactions on the blockchain for that wallet. Most current cryptocurrency implementations, such as Bitcoin (Nakamoto, 2008), rely on proof-of-work approaches to process transactions and generate new currency, which can only be done by adding new blocks to the emergent blockchain, which requires searching for nonces that will result in allowable hash values.

By enforcing mining (proof-of-work), the Bitcoin DLS attempts to mitigate the Sybil attack (Douceur, 2002), also called the 51 per cent attack by some authors, by requiring a computational investment for every transaction. With this, a malicious adversary must make an investment of at least half the compute (mining) power of the network to accomplish the Sybil attack. If a Sybil attack is successfully carried out then, from then on, the DLS is completely compromised and the emergent blockchain can be controlled by the adversary. In such a case, the adversary can dictate which new transactions are confirmed and who receives new currency. Furthermore, there is probably no reliable method to detect such attack. This is a foundational pillar of the Bitcoin approach and one of the major drawbacks of a public non-permissioned DLS. This is because, currently, the Sybil attack is not preventable unless the identity of agents can be verified and users are prohibited from creating multiple identities or agents (Douceur, 2002).

In Bitcoin, as with other cryptocurrencies, users can mine collaboratively by creating *mining pools*. Collectively, compute nodes in a mining pool discover target hash values at a more constant rate. The profits of the creation of new bitcoins can be shared across the pool. Mining pools also add the risk of two new adversarial types: malicious pool operators, and malicious pool members (Rosenfeld, 2011). Malicious pool operators could attempt to perform a Sybil attack on the network using the combined resources of their pool. In the case of Bitcoin, high risk situations have arisen where a prominent mining pool controlled the majority of the agents in the network with the resulting risk to the system. Monetary and social pressure have since resulted in a reduced presence of such pools (Goodin, 2014). This is an area where current DLSs must be greatly improved. For widespread adoption, the system must provide proof that no malicious adversary will be

able to singularly control the emergent transaction ledger (Eyal and Sirer, 2014). Malicious pool members could contribute to the apparent computational power of a mining pool but attempt to destabilize the pool in the long run. This may be accomplished by hopping from pool to pool or joining multiple pools to increase expected earnings or by temporarily withholding collaboratively mined blocks to sabotage the effectiveness of some of the pools (Rosenfeld, 2011). These two new kinds of attacks, enabled by the presence of mining pools, could be mitigated by dis-incentivizing mining pool cooperation. Current common cryptocurrencies such as Bitcoin and Ethereum do not deter mining pools. However, proposals to do so exist (Eyal and Sirer, 2014, Velner *et al.*, 2017). However, the threat to most currently proposed solutions is the lack of assurance that all agents are distinct and are not colluding. Assuring this in an open network and with a non-permissioned blockchain would be practically impossible.

For Bitcoin, modifying the system to deterring mining pools would require a hard fork of the Bitcoin blockchain and the creation of a new DLS (Eyal and Sirer, 2014). In the case of Ethereum, rewarding users for sabotaging their pool (Velner *et al.*, 2017) would require use of additional contract-driven functionality atop the Ethereum blockchain (Buterin, 2013). Either of these approaches may accomplish the goal of changing the overall return on investment when joining a mining pool. Malicious pool operators attempting a Sybil attack (Douceur, 2002) would again be required to compete against the entire network for resource parity. Malicious pool members could be incentivized to short-circuit their pool for personal gain which would in turn deter members to join mining pools. This is where permissioned and authenticated DLSs offer advantages over non-permissioned systems such as Bitcoin and Ethereum.

The proof-of-work approach of Bitcoin and Ethereum has been stated as able to mitigate the Sybil attack. This would be accurate only under the assumption that all network identities have access to similar computing resources and that agents and identities are mapped one to one. Neither of these assumptions can be assured by these systems since users can join with as many agents as they wish to create. Organizations with vast amounts of financial and computing resources, such as state-backed organizations, could break both such assumptions. As mentioned before, another drawback of the proof-of-work approach is the exponential increase in the cost and time per block and hence per transaction.

As demonstrated by Douceur (2002), user-identity certification is very likely the only complete solution to the Sybil attack. Other proposed solutions that do not verify user's identities and establish adequate access control on the system exhibit much higher risks of failure and fraud. Financial entities are required to evaluate risks and implement mitigation strategies to minimize risks (Walch, 2017). This means that an external means for validating identities and matching users to identities would be needed if DLSs are to be in widespread use in the business-to-business and financial worlds. This is not a high barrier since in most countries financial institutions are already required to verify and validate the identity of their customers. It does mean that identity verification agencies will need to be initially involved during account creation before users can join a trusted DLS. This fact most likely indicates that business-class DLSs will most likely be a hybrid system, rather than a purely centralized or purely distributed system. It also means that business-class DLSs will most likely be permissioned. This presents a threat to the use of DLSs as an essential component of large Internet of things systems. Hence, trust in DLTs is an area that needs further investigation. For example, DLTs in which user trust is assigned or gained and lost rather than assumed or not present should be investigated and evaluated.

7. Challenges with smart contracts

Another common application of the internet is offloading computation or storage tasks to cloud or third-party providers. Currently, cloud solutions such as Amazon Azure, Google Cloud and Digital Ocean, among many others, enable remote computation and storage. These services are usually made available at different layers such as IaaS, SaaS or Applications as a Service. The Ethereum project (Buterin, 2013, Wood, 2016) seeks to provide programmable computation, albeit possibly not Turing complete, atop a DLS. Transactions on Ethereum may contain, in addition to currency assignment and transfer, program code called a *smart contract* (Ethereum, 2016). Smart contracts, first described by Szabo (1997), establish a digital relationship between two parties. In Ethereum, a smart contract is a transaction and hence recorded on the blockchain. The code contained in an Ethereum smart contract supports the relationship described by Szabo (Szabo, 1997; Buterin, 2013). Smart contracts carry the input values required by the computation and an address to which currency may be sent. The result of a smart contract's computation is then reflected in the *emergent* blockchain and the current state of the Ethereum Virtual Machine (Wood, 2016). In the Ethereum system, similarly to the Bitcoin system, transactions and smart contracts carry computation and agreement costs. Whereas Bitcoin only allows external entities to interact with the blockchain, Ethereum allows smart contracts, and hence computations, to do so as well (Buterin, 2013). Ethereum contracts may also perform transactions against each other. In other words, smart contracts are automated or event-triggered transaction agents.

Ethereum smart contracts are implemented by programming code. As such, they may contain unintended or malicious flaws (Delmolino *et al.*, 2015). This may allow malicious actors to control other users' actions or results. In June of 2016, the virtual DAO implemented as a smart contract in Ethereum, which provided crowdfunding capabilities, was attacked by exploiting a vulnerability within the smart contract program code that allowed a malicious user to siphon cryptocurrency from the virtual organization's funds (Buterin, 2016a). After the vulnerability was discovered, it was mitigated. However, the only recourse to recover the lost cryptocurrency was to agree to a modification of the Ethereum blockchain by eliminating all the malicious transactions after a certain block (Buterin, 2016b). Eighty-five per cent of Ethereum network members arrived at a consensus on the new blockchain by mining the new branch (Buterin, 2016b). This fact clearly disproves statements that the emergent blockchain in a DLT system is immutable and that transactions are unchangeable. The execution of this event is technically like the case of the Sybil attack described earlier (Goodin, 2014). As a possible analogy with the business world, such event could be argued similar to a hostile takeover for the undoing of a previously made contract that was considered invalid or fraudulent by the majority of the stockholders, along with all subsequent related contracts and transactions, which had also been previously agreed by a majority of the shareholders. The legal, ethical and technical implications of DLSs, and their rules of conduct, must be studied and analyzed further before these systems are in widespread use in the business world.

Contracts as a Service (CaaS) could be considered a service offering in between IaaS and SaaS. Currently, for IaaS or SaaS, and within the realm of a centralized and trusted service provider, users provide their sensitive information for computation as necessary. This is done under the assumption that the service provider will not accidentally, or due to an external or internal malicious act, share the private data used by, or resulting from, the computation with unauthorized parties. Other examples of distributed storage and computation, in addition to DLTs with smart contracts, such as Ethereum, include peer-to-

peer file systems (Benet, 2014) and the Urbit distributed computation system, which is in early research stages (Yarvin *et al.*, 2016).

The challenges that must be addressed before market-scale adoption of smart contracts or Distributed Contracts as a Service (DCaaS), in addition to all the challenges of DLTs and DLSs, are: ensuring the availability of computational power; ensuring confidentiality and privacy, even under the assumption of adversary actors; ensuing the correctness security and trustworthiness of the contract execution platforms; and ensuring the correctness and compliance to rules of each and all smart contracts before they are incorporated into the system.

In a distributed environment, a malicious user may offer contract storage and computation services for the purposes of stealing user data or manipulating results. A benign user may unknowingly release confidential data or calculate incorrect results. The second challenge will need to be solved by markets, paid for by government entities, or by win–win consortium agreements in the case of business-to-business systems. The third challenge could be addressed by recent technologies for computation on encrypted data such as zero-knowledge proofs or fully homomorphic encryption. The fourth challenge must be addressed by the application of: research with critical analysis and repeatable experiments, sound software engineering practices, selective application of formal methods, and thorough and exhaustive testing. The fifth challenge would need fundamental research on domain-specific languages and computational correctness. Additional and multidisciplinary research is needed in all these areas.

8. Advances with encrypted computation

Fully homomorphic encryption provides a method for performing computation on encrypted data (Damgård and Jurik, 2000, Gentry, 2009, Gentry *et al.*, 2013). Homomorphic encryption, while less computationally efficient than direct computation on cleartext (non-encrypted data), has the potential to enable Computation as a Service or smart contract offerings, both Distributed (DCaaS) and Cloud-based (CaaS), while maintaining the confidentiality and privacy of both the input and output data. The first fully described homomorphic encryption technique, designed by Gentry (2009, 2013) in a dissertation work, is based on the *shortest vector* and *shortest independent vector* problems of ideal lattices, which are mathematical constructs. These problems are based on finding the shortest vector within a lattice, in a given vector space, given the basis and the norm (Ajtai, 1996). This technique provided operations for encrypted addition and multiplication, a requirement for encrypted computation. Gentry's original approach incurred an error value that accumulated as computations were performed and this forced a periodic reset of the computation to keep errors within a tolerable margin (Gentry, 2009).

Newer homomorphic encryption techniques, also pioneered by Gentry and colleagues, are based on the learning with errors problem, which offers similar guarantees as lattice-based techniques with less computational overhead (Gentry *et al.*, 2013; Regev, 2005). The approximate eigenvector technique provides operations for encrypted addition and multiplication with computational complexity similar matrix addition and multiplication (Gentry *et al.*, 2013). Here, the secret key is a vector that is the approximate eigenvector of the ciphertext matrix. By including a noise parameter and saying the secret key is only an approximation of the eigenvector the problem is reduced to the learning with errors problem (Gentry *et al.*, 2013).

The correct combination of encrypted computation with advanced distributed computation and storage approaches, such those enabled by DLTs or DLSs, could enable a new class of cloud- and fog computing-based markets for data storage and computation services. In such markets, both data storage and computation would be distributed and brokered, while the availability, integrity and confidentiality of the data are also ensured, even under the adverse effects of data loss due to cyberattacks on service providers.

9. Conclusion

Adequately addressing data availability, integrity and security in an efficient manner is an essential aspect in most computing applications. New DLTs or DLSs appear to be a promising approach to addressing data availability and integrity for logbook-style data such as accounting and financial ledgers and asset ownership, tracking, and provenance. If investigated adequately and implemented correctly, safely and securely, these technologies have the potential for a paradigm shift on data storage and processing in many information system areas which have been traditionally a stronghold of centralized systems based on paper or relational databases.

In this article, we introduced blockchains and DLSs and described their essential and desired emergent properties. We clarified what are intrinsic properties of the technology versus what are desired but emergent properties of a DLS. Emergent system properties are notoriously hard to prove present and correct. We described the challenges that DLTs will need to address before being trustworthy enterprise wide. We also described how the combination of encrypted computation and asymmetric cryptographic techniques with DLTs could lead to the creation of new distributed storage and contracting markets based on the blockchain and smart contracts. However, we found that much more fundamental and practical multidisciplinary research is needed to ensure the security and trustworthiness of the developed systems before they are in widespread use.

References

Ajtai, M. (1996), "Generating hard instances of lattice problems", *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of computing, ACM, Philadelphia, PA*, pp. 99-108.

Benet, J. (2014), "IPFS - content addressed, versioned, P2P file system", CoRR, abs/1407.3561, available at: http://arxiv.org/abs/1407.3561

Buterin, V. (2013), "A next generation smart contract and decentralized application platform", available at: www.the-blockchain.com/docs/Ethereum-white-paper-a-nextgeneration-smart-contract-and-decentralized-application-platform-vitalik-buterin.pdf

Buterin, V. (2016a), "Critical update re: DAO vulnerability", available at: https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability/

Buterin, V. (2016b), "Hard fork completed", available at: https://blog.ethereum.org/2016/07/20/hard-fork-completed/

Damgård, I. and Jurik, M. (2001), "A generalisation, a simplification and some applications of paillier's probabilistic public-key system", in Kim, K. (Ed.), *Public Key Cryptography*, Lecture Notes in Computer Science, Springer, Berlin, Vol. 1992, doi: https://doi.org/10.1007/3-540-44586-2_9.

Delmolino, K., Arnett, M., Kosba, A., Miller, A. and Shi, E. (2015), "Step by step towards creating a safe smart contract: lessons and insights from a cryptocurrency lab", available at: https://eprint.iacr.org/2015/460.pdf

Douceur, J.R. (2002), "The Sybil attack", *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS-2002), Lecture Notes in Computer Science, Springer-Verlag, Berlin*, Vol. 2429, pp. 251-260.

Ethereum, F. (2016), "Introduction to smart contracts", available at: https://solidity.readthedocs.io/en/develop/introduction-to-smart-contracts.html

Eyal, I. and Sirer, E.G. (2014), "It's time for a hard bitcoin fork", available at: http://hackingdistributed.com/p/2014/06/13/in-ghash-bitcoin-trusts/

Fischer, M.J., Lynch, N.A. and Paterson, M.S. (1983), "Impossibility of distributed consensus with one faulty process", *Proceedings of the 2nd ACM SIGACT645 SIGMOD Symposium on Principles of Database Systems, Association of Computing Machinery, New York, NY*.

Gentry, C. (2009), "A fully homomorphic encryption scheme", Ph.D. dissertation thesis, Standford University, Standford.

Gentry, C., Sahai, A. and Waters, B. (2013), "Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based", Cryptology ePrint Archive, Report 2013/340.

Gervais, A., Karame, G.O., Wst, K., Glykantzis, V., Ritzdorf, H. and Capkun, S. (2016), "On the security and performance of proof of work blockchains", *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS-2016), Association of Computing Machinery, New York, NY*, pp. 3-16.

Goodin, D. (2014), "Bitcoin security guarantee shattered by anonymous miner with 51% network power", available at: https://arstechnica.com/security/2014/06/bitcoin-security-guarantee-shattered-by-anonymous-miner-with-51-network-power/

Gupta, M. (2017), *Blockchain for Dummies, IBM Limited Edition*, John Wiley and Sons, Hoboken, NJ, available at: https://public.dhe.ibm.com/common/ssi/ecm/xi/en/xim12354usen/XIM12354USEN.PDF

Hyperledger, F. (2017), "Hyperledger: Linux foundation projects", available at: www.hyperledger.org/

Intel, C. (2017), "The second coming of blockchain", available at: https://software.intel.com/en-us/blogs/2017/02/14/the-second-coming-of-blockchain

Nakamoto, S. (2008), "Bitcoin: a peer-to-peer electronic cash system", available at: https://bitcoin.org/bitcoin.pdf

NIST (2015a), "Secure hash standard", NIST FIPS 180-4, doi: 10.6028/NIST.FIPS.180-4, available at: www.nist.gov/publications/secure-hash-standard

NIST (2015b), "SHA-3 standard: permutation-based hash and extendable-output functions", NIST FIPS 202, doi: 10.6028/NIST.FIPS.202, available at: www.nist.gov/publications/sha-3-standard-permutation-based-hash-and-extendable-output-functions

Norton, S. (2016), "CIO explainer: what is blockchain?", *The Wall Street Journal Blog*, available at: https://blogs.wsj.com/cio/2016/02/02/cio-explainer-what-is-blockchain/

Popper, N. and Lohrmarch, S. (2017), "Blockchain: a better way to track pork chops, bonds, bad peanut butter?", *The New York Times*, News Article, available at: www.nytimes.com/2017/03/04/business/dealbook/blockchain-ibm-bitcoin.html

Regev, O. (2005), "On lattices, learning with errors, random linear codes, and cryptography", *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, ACM*, pp. 84-93.

Rosenfeld, M. (2011), "Analysis of bitcoin pooled mining reward systems", *CoRR*, abs/1112.4980, available at: http://arxiv.org/abs/1112.4980

Santander, B. (2015), "Santander becomes first UK bank to introduce blockchain technology for international payments with the launch of a new app", Santander News Room, available at: www.santander.com

Schneider, J., Blostein, A., Brian, L., Kent, S., Groer, I. and Beardsley, E. (2016), "Blockchain: putting theory into practice", *Profiles in Innovation Report*, The Goldman Sachs Group.

Stallings, W. and Brown, L. (2015), *Computer Security: Principles and Practice*, 3rd ed., Pearson Education, London.

Szabo, N. (1997), "Formalizing and securing relationships on public networks", *First Monday*, Vol. 2 No. 9.

Velner, Y., Teutsch, J. and and Luu, L. (2017), "Smart contracts make bitcoin mining pools vulnerable", available at: https://eprint.iacr.org/2017/230.pdf

Walch, A. (2017), "Should public blockchains serve as financial market infrastructures?", *Handbook of Digital Banking and Internet Finance*, Elsevier, Amsterdam, Vol. 2, available at: https://ssrn.com/abstract=2879239

Wood, G. (2016), "Ethereum: a secure decentralised generalised transaction ledger", available at: http://gavwood.com/paper.pdf

Wüst, K. and Gervais, A. (2016), "Ethereum eclipse attacks", ETH Library, available at: http://e-collection.library.ethz.ch/eserv/eth:49728/709eth-49728-01.pdf

Yarvin, C., Monk, P., Dyudin, A. and Pasco, R. (2016), "Urbit: a solid-state interpreter", available at: www.urbit.org

Further reading

Lamport, L., Shostak, R. and Pease, M. (1982), "The byzantine generals problem", *ACM Transactions on Programming Languages and Systems (toplas)*, Vol. 4 No. 3, pp. 382-401.

Corresponding author

Daniel Conte de Leon can be contacted at: dcontedeleon@ieee.org