Received: 23 November 2018 | Revised: 1 February 2019 | Accepted: 2 February 2019

DOI: 10.1002/itl2.93

LETTER

WILEY

Research challenges and opportunities in blockchain and cryptocurrencies

Qusay H. Mahmoud[1] | **Michael Lescisin**[1] | **May AlTaei**[2]

[1]Department of Electrical, Computer, and Software Engineering, University of Ontario Institute of Technology, Oshawa, Ontario, Canada
[2]College of Technological Innovation, Zayed University, Abu Dhabi, UAE

Correspondence
Qusay H. Mahmoud, Department of Electrical, Computer, and Software Engineering, University of Ontario Institute of Technology, 2000 Simcoe Street North, Oshawa, Ontario L1G 0C5, Canada.
Email: qusay.mahmoud@uoit.ca

The blockchain is the underlying technology of the Bitcoin cryptocurrency, and it has created much excitement in the technology and research communities. A blockchain is a distributed ledger collectively maintained by a peer-to-peer network of participants who in Bitcoin are known as miners. This key innovation enables cryptocurrencies such as Bitcoin to operate in a decentralized manner with no intermediaries such as financial institutions. But the blockchain can be used to record things other than cryptocurrency transactions. While many of the concepts of Bitcoin build on what have been around since the 1980s and 1990s, the designer(s) of it have made important assumptions that make it work along with the use of an incentive protocol, leading to a major breakthrough from traditional academic thinking. In this paper, we present the state-of-the-art of blockchain and cryptocurrencies along with research challenges and opportunities that would be of interest to researchers getting into this exciting field.

KEYWORDS
bitcoin, blockchain, cryptocurrency, distributed systems

1 | INTRODUCTION

A blockchain is a distributed ledger forming a distributed consensus on a history of transactions. The idea of blockchain first emerged into the public sphere with the Bitcoin P2P cryptocurrency, however, its applications go far beyond the scope of the financial sector. Specifically, the blockchain is extended into the broader scope through the use of *smart contracts*. A *smart contract* applies the *distributed consensus* property of blockchain to create enforceable contracts for any type of digital asset. A *smart contract* is essentially a small program which all participants on the blockchain network execute and perform subsequent actions based on the result of the *smart contract* execution. Given that the *smart contract* execution is deterministic, a distributed consensus can be formed.

It has been said that *blockchain* will do to middle and back office functions what the Internet and the Web has done to front office functions—that is, automate functions thus bringing efficiency and new business opportunities.[1] Regardless of the application of the *blockchain* technology, the underlying theory is relatively straightforward—each transaction is a cryptographically signed message that consumes inputs (transfer of value from transaction initiator) and creates new outputs (transfer of value to another party). In order for each node on the blockchain network to verify if a transaction is legitimate, all transactions are broadcasted over the P2P network. The simple broadcasting of transactions, however, is an incomplete solution. Due to network propagation delays and nodes going on and off line, each node will not hold an identical copy of the history of transactions. To solve this problem, the distributed system design must satisfy two properties; *the frequency of transaction activity must be limited to something much lower than propagation delays* and *each transaction activity must make reference to the previous transaction activity* thus forming a *chain* of history. These two properties are implemented in *blockchain* systems by *(1) clustering broadcasted transactions into blocks* and by *(2) making each block reference the previous block*. By following these rules, there is still one unanswered question—*how can the distributed network form a consensus as to what block of clustered transactions is the valid one?*. To solve this issue, P2P nodes (or *miners*) *compete* to find the next *valid* block. In addition to verifying that all transactions come from addresses that actually have the available currency, and that the current block makes reference to the previous block, an additional criterion needs to be satisfied—that is, a block-related *difficult to solve, easy to verify* computational problem needs to be solved. This computational problem is what limits the rate of valid blocks from appearing on the network and thus a distributed consensus can be formed on the current history of transactions.

To this end, the research contribution of this paper is 2-fold: analysis of the state-of-the-art in blockchain and cryptocurrencies, along with investigation of associated research challenges and potential solutions.

The rest of the paper is organized as follows. Section 2 discusses the definition of a blockchain and the scope of problems suited to solve, and the *economics* of *mining* on a blockchain network. Section 3 forms the core of the research contribution of this paper discussing the current *research challenges* on the various scopes of blockchain development. The paper is concluded in Section 4.

2 | BLOCKCHAIN AND CRYPTOCURRENCIES

The concept of *cryptocurrency* was the first, and currently is the most popular use of *blockchain* technology. The idea of *cryptocurrency* is simply the transfer of value between parties using only cryptography, and not central financial institutions, to ensure authenticity. Therefore in *cryptocurrency*, the ledger of a centralized bank is simply replaced by the distributed ledger that is the *blockchain*.

The applications of *blockchain*, however, go far beyond implementing cryptocurrencies. While the concept of a distributed ledger is quite understandable, its actual secure implementation and its applications provide groundbreaking results.

To understand the breakthrough that was made with the creation of the *blockchain* one must first understand the *Byzantine Generals Problem*.[2] The *Byzantine Generals Problem* is a classical problem in distributed systems illustrated by several armies converging to attack a castle. The castle may only be conquered if *all* armies attack at the same time. Consider a naive solution where the *leading* army uses a messenger to instruct all other armies to attack at a given time. The messenger could be captured while in transit and thus the *attack* message would never be delivered. To confirm that a message was delivered, the sending party could ask for a *message acknowledgement* but the same problem is encountered again as the deliverer of the *message acknowledgement* could also be captured. Therefore, a *consensus* needs to be reached certifying that *(1) the transmitter of the attack message knows that all other armies have received this message* and *(2) every army that has received this message can confirm that all other armies have received this message*. Now suppose that instead of simply sending a messenger, the *leading army general* posts his attack message to a *blockchain*. The attack is scheduled to be at a time that is 12 hours from now. The *proof of work* used on this *blockchain* is such that if all armies work on solving the problem at the same time, it will take approximately 10 minutes for the first solution to appear. Once the general who has sent the *attack message* sees valid proof of work solutions appearing approximately every 10 minutes, he can be assured that all other armies have received this message as it would be impossible for any fewer amount of armies to be producing valid *proof of work* solutions at this rate. Similarly, every other army can be *highly confident* that every other army has seen the *attack message* given the rate at which *proof of work* solutions are being produced.

The same problems that are described by the *Byzantine Generals Problem* are also applicable to the idea of a *distributed ledger*. Just as the *blockchain* solution to the *Byzantine Generals Problem* ensures that all parties know that all other parties have seen a message, so can this method also be used to verify that all parties agree on the current state of a ledger. Given that we now can have a *consensus* on a *distributed ledger* we may now use this ledger for a wide variety of applications. In a simple case, we may use it as a value transfer system thus implementing *cryptocurrency*. In more complicated systems, we may use it to establish consensus of the state of *digital assets*, thus implementing *smart contracts*.

2.1 | Proof of work and mining

Mining is the process of finding the next valid block to be placed on the blockchain (Figure 1). The objective is to *competitively* solve computationally difficult problems in order to limit the rate at which new blocks are created. Nodes on a *blockchain* network are incentivized to participate in *mining* as if they are the first to obtain a valid block, the distributed network rewards them with an award of cryptocurrency. This award may be sourced from inflation, known as a *block reward* or it may be collected from transaction fees. Most cryptocurrencies are designed so that all coins in circulation are generated from *block rewards* and once all coins have been generated, miners are then incentivized through *transaction fee rewards*. The network and currency stability implications of transitioning to an exclusively *transaction fee rewards* based scheme is still an ongoing research topic.[3]

Through the simple laws of *supply and demand*, due to the *work* required to generate cryptocurrency coins, these digital coins obtain their economic value. In context of the *Byzantine Generals Problem*, the *proof of work* also ensures that a large amount of peers have seen and verified the block given the rate at which valid blocks are produced. *Proof of work* also defends the distributed network against *Sybil attacks* by making it expensive, both in terms of computational hardware as well as electricity consumption, to create large amounts of false identities.

3 | RESEARCH CHALLENGES

3.1 | Permissioned and permissionless blockchain

Most commonly, the term *blockchain* is used to refer to a *public* or *permissionless* distributed ledger. This however, is not the only use of the technology. If users are willing to accept some *centralization*, then the computationally expensive, and energy consuming *proof of work algorithm* can be replaced by a list of *trusted* users who may sign blocks as valid. This centralized/decentralized *hybrid* approach can greatly simplify *value transfer* and *consensus* between trusted parties. To understand the application of *permissioned blockchains* consider the following example.

Suppose there are many tasks to be done on a construction project. For this project, the owner hires multiple contractors. Suppose there are a bricklayer, an electrician, and a plumber. After agreeing upon a construction timeline and dependency graph (eg, basic frame must be in place before plumbing is installed) all parties agree that they trust the project owner as the *payment certifier*—that is, the party who has the authority to declare that the work was done correctly. By following the rules of this contract, the contractors begin their work starting at the root dependencies. As the project progresses along, the role of the *permissioned blockchain* comes into play. Once the basic brick structure has been created, the owner creates a signed transaction on the *permissioned blockchain* stating that the basic foundation work has been completed. The bricklayer can then use the existence of this signed transaction to prove that they should be paid. Furthermore, the electrician and plumber are now aware that the owner has agreed that basic foundation work is now completed and they now may begin their construction tasks.

The *construction project* enterprise blockchain is merely a *simple* example of how blockchain can be leveraged to facilitate commerce. In reality, business structures and models can vary greatly and therefore a *flexible* framework for implementing blockchains is necessary. Specifically, *privacy* is often a common

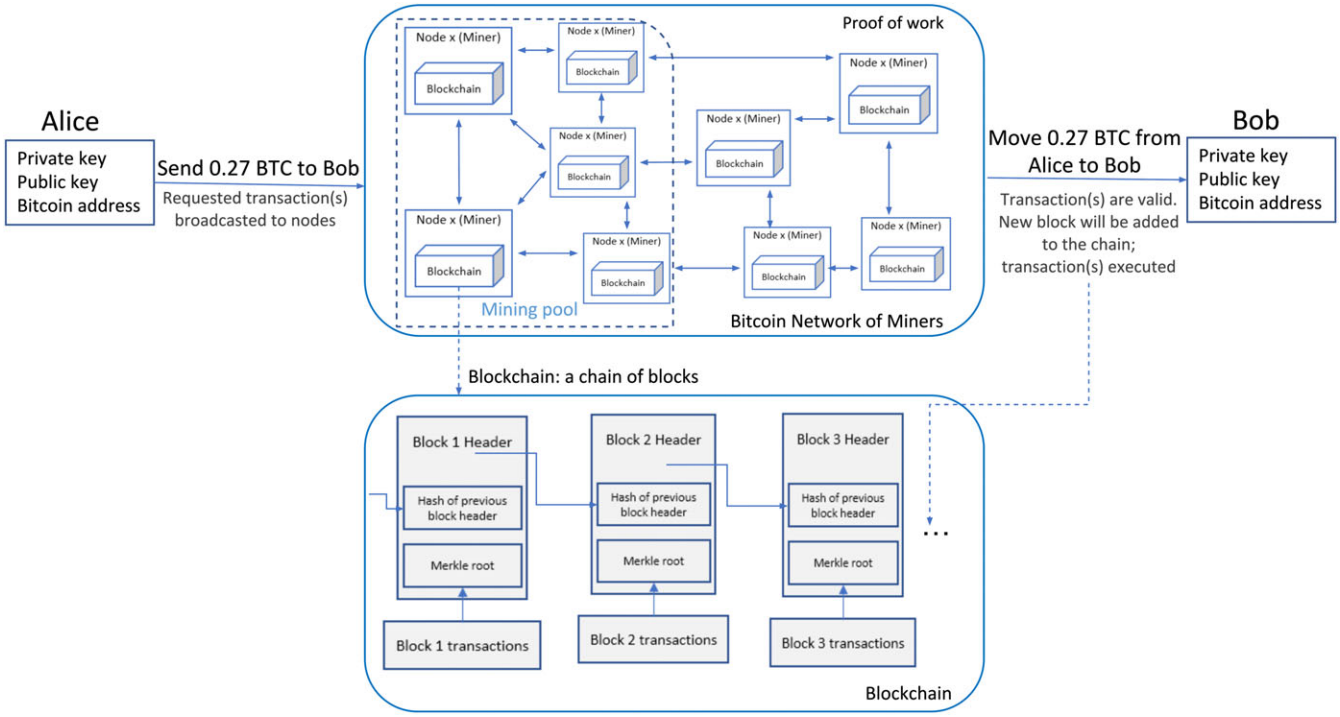

FIGURE 1 Distributed consensus is achieved by proving that enough parties agree on the ledger state by expending computational efforts through the process of block mining

concern within business relations. *Hyperledger Fabric*[4] is designed to handle both of these concerns as it is a *framework* for implementing blockchains with the support for *channels*. A *channel* in *Hyperledger Fabric* is simply a *ledger* that is distributed *only* to a select group of parties. It is therefore comparable to a channel in an Internet messaging service where only specific users have access permissions.

3.2 | Consensus mechanisms

Consensus in *private permissioned* blockchains is a relatively simple process. For example, in a *Hyperledger Fabric* blockchain that is private between a group of business partners, all parties can come to a consensus on whom they trust to approve transactions. Similarly in *Ripple*,[5] consensus is achieved by each server maintaining a *Unique Node List (UNL)* composed of other servers which said server trusts that their votes on a block's validity will not be used to defraud the network. These techniques for achieving *consensus* in a *trusted* network, unfortunately, cannot be applied to *untrusted* networks.

The *consensus mechanism* used in *public blockchain* systems is still an ongoing research problem. For *proof-of-work* schemes such as Bitcoin, a large amount of electricity is used for operating computers to solve *difficult to solve, but easy to verify* computational problems which have no useful application other than securing the blockchain. To alleviate this problem, research has been directed two ways; *(1) eliminating the need for proof of work* and *(2) making the result of the proof of work beneficial to society*.

PeerCoin[6] is an example of a *blockchain* application which has as a goal the elimination of computationally expensive *proof of work* problems. To accomplish this while still maintaining secure distributed consensus, the idea of *proof of work* is replaced with the idea of *proof of stake*. In a *proof of stake* algorithm, a peer is chosen at random to sign the current block. The probability that a given peer will be chosen is proportional to the amount of currency which they own—thus, thwarting *Sybil attacks* where an adversary could inexpensively create many identities and thus have a large probability of being chosen to sign the latest block.

PrimeCoin[7] is an example of a *blockchain* application which has the goal of making the *difficult to solve, easy to verify* problem useful to society. The *proof of work* problem used by PrimeCoin is finding a *Cunningham* or *Bi-twin* chain of prime numbers of a given length. Unlike *Bitcoin's*, *Hashcash-based* proof of work algorithm, the Cunningham or Bi-twin chains of prime numbers found by PrimeCoin miners has useful applications to mathematicians.

In addition to the large amounts of electricity that are used for *mining proof-of-work* based cryptocurrencies, the extremely high levels of computational performance needed for profitable cryptocurrency *mining* creates a very high barrier to entry to become a miner and thus the cryptocurrency system that was designed to be *decentralized* tends towards *centralization*.

Consider the evolution of *Bitcoin* mining. First, there was *CPU mining*. Then, there was *GPU mining*. Followed by that was *FPGA mining*. After that came *ASIC mining*. As of current, *Bitcoin* can now only be mined profitably on *large farms of ASICs*. Therefore, it can be said that *Bitcoin* is moving towards increased *centralization* as more and more of the network's hashing power is being concentrated to a small number of parties.

ASICs are relatively simple to design for *Bitcoin* mining as they only need to do one thing very quickly and efficiently—that is, calculate SHA256 hashes. As the amount of binary storage elements required to calculate a SHA256 hash is relatively small, many of these circuits can be built into a single IC thus achieving a high hash rate while consuming minimal amounts of energy. The cost and energy overhead associated with implementing a *Turing-complete* machine is not necessary when the integrated circuit only needs to calculate SHA256 hashes.

One of the first *altcoins* to tackle the idea of ASIC resistance, was *Litecoin*. *Litecoin* uses the *scrypt key derivation function* in its proof of work algorithm instead of SHA256 as is used in *Bitcoin*. Unlike *SHA256*, *scrypt* requires large amounts pseudorandomly generated strings to be held in memory and accessed at random. Due to this memory intensive requirement for *scrypt*, it more difficult to design an ASIC for this type of *key derivation function* than it is for simple SHA256 hashes.

As can be seen from an overview of the blockchain *proof-of-work* techniques, there is still much left to be desired. *Proof-of-stake*, although low in terms of energy consumption, encourages *hoarding* and discourages secure *offline* storage of *private* wallet keys. *Proof-of-work* is energy intensive and, with the exception of *PrimeCoin*, does not produce a useful knowledge result. An ongoing research challenge for *proof-of-work* is to find a set of *scientific computing* problems which are difficult to solve *but* easy to verify. Many *scientific computing* problems can only be verified by reproducing the steps used to obtain the solution. The ongoing research challenge for these types of problems, therefore, is to ensure that a valid solution can be computed *without* requiring all peers to compute the *computationally intensive* solution.

3.3 | Scalability

Given that a *blockchain* is an *append-only* data structure, its ever growing size is a concern when considering system *scalability*. By the end of September 2018, the *Bitcoin* blockchain was 185 gigabytes in size.[8]

One approach to lowering the disk storage requirements for running a full *blockchain* node is *blockchain pruning*. To properly implement *blockchain pruning* we must remember that the hash of the latest block *must* be representative of the current state of all users' wallet balances. To illustrate *blockchain pruning* with a simple example, consider the following. We have users *A*, *B*, and *C*. All users have a wallet balance of 10 coins. The first block is placed on the *blockchain* representing a transfer of 1 coin from *A* to *B*. The current wallet state of the network is that *A* has 9 coins, *B* has 11, and *C* has 10. Another block now appears on the *blockchain* representing a transfer of 1 coin from *B* to *C*. The current wallet state of the network is now *A* has 9 coins, *B* has 10, and *C* has 11, thus, the transfers from *A* to *B* and *B* to *C* can be replaced by a single transfer from *A* to *C*.

Hard drive space required for an ever-growing blockchain is not the only scalability concern with *blockchain* systems. The processing rate of *blockchain* transactions is also limited by the processing power and network bandwidth of the P2P nodes. Currently, the *Bitcoin* network can process transactions at a maximum rate of 7 per second, a figure that is approximately 1000 times smaller than the peak capacity of the VISA transaction processing network.[9] To improve the transaction processing rate of *blockchain* based systems, the idea of a *lightning network* has been proposed.

A *lightning network* is a graph whose *vertices* are consumers/producers and whose *edges* are referred to as *payment channels*. A *payment channel* is a blockchain listed balance sheet describing a programmatically enforceable contract of a transfer of funds between two parties.[10] To understand how *payment channels* work, consider the following example. Suppose *Bob* frequently rides the *AnyTown Metro*. The fare for riding this transit system is *1 coin*. In the initial creation of the *payment channel*, Bob allocates *100 coins* and places them in a *multi-signature wallet* thus requiring both the signatures of himself as well as *AnyTown Metro* in order for these *100 coins* to be accessed. Bob also creates a new balance sheet describing that *100 coins* will be sent to him and zero to *AnyTown Metro*. This balance sheet is signed by both parties but is kept *private*. If this signed balance sheet were to be released to the blockchain, *Bob* would reclaim his *100 coins* and the payment channel would close. When *Bob* pays for using the *AnyTown Metro*, a new balance sheet is created dictating that *Bob* will receive *99 coins* and *AnyTown Metro* will receive *1 coin*. This procedure repeats itself as long as *Bob* continues to use *AnyTown Metro* and the payment channel remains active. Now, suppose *Bob* moves away and the balance sheet shows that *Bob* will receive *25 coins* and *AnyTown Metro* will receive *75 coins*. *Bob* can close the *payment channel* by broadcasting this balance sheet to the *blockchain network* and he will receive back his *25 coins* while *AnyTown Metro* receives their *75 coins*. Lastly, after every update to the balance sheet, the *private* keys used to sign the previous balance sheet are exchanged thus allowing any party to the *payment channel* to refute the other party's fraudulent claim of an older (and more favorable) balance sheet being the current version.

Lightning networks have, however, been criticized for bringing *centralization* to cryptocurrency networks. In order for a payment to be routed over a *lightning network*, all links in the network must have sufficient capacity—that is, funds on their balance sheets that are greater than or equal to the value of the payment that is to be sent. If the payment to be sent over *lightning network* is large, then only a small set of routes through the *lightning network* will be available, thus increasing centralization.

Another solution to *blockchain scalability*, ironically, is to eliminate the need for a blockchain. Specifically, IOTA[11] establishes *distributed consensus* by using a *directed acyclic graph (DAG)* instead of a blockchain. In order to add a *transaction block* to the IOTA DAG, the new block must simply reference two valid *edge* blocks. An *edge* block is simply a *transaction block* which makes reference to other blocks but has not yet been referenced. A transaction block in IOTA is *confirmed* each time the block moves away from the *edge* of the network due to being referenced by new blocks. IOTA has been criticized, however, as, if at least 33% of the network participants are controlled by an adversary, blocks could be illegitimately added to the DAG. Furthermore, this *33% attack* is of great concern to IOTA developers as the network is still relatively small. To guard against *33% attacks*, *coordinator nodes*, which are *privileged nodes* controlled by the IOTA developers have been created, thus making the system not completely decentralized. Therefore, if a *33% attack* were to occur, the *coordinator nodes* would be able to exercise their authority to stop the attack.

3.4 | Security

The *blockchain* and *cryptocurrency* spheres have given rise to a number of innovative security measures which provide resistance to theft of *cryptocurrency* but have also shown promising application to other areas of computer security.

Under the simplest use of a cryptocurrency such as *Bitcoin*, if a wallet's private key is compromised, then so is all the cryptocurrency held in that wallet. Given this utmost importance of ensuring that a wallet's private key remains private, two innovative solutions have emerged for the protection of the confidentiality of the private key; *hardware wallets* and *paper wallets*.[9]

In a *hardware wallet*, the private key is held on an integrated circuit and is never transmitted to any device. Therefore, when making a purchase using cryptocurrency stored on a hardware wallet, the potentially compromised host sends an unsigned transaction to the hardware wallet. The owner of the hardware wallet then verifies that the transaction is legitimate, and if the owner approves the transaction, the hardware wallet generates the appropriate cryptographic signature and returns the signed transaction to the requesting host where it is then sent to the blockchain network.

In a *paper* wallet, a paper note holds both the *private* and *public* keys for a *cryptocurrency wallet*. Therefore, by using a *paper* wallet one can create their own denominations of a cryptocurrency simply by loading the desired amount to the wallet address defined by the *public-private keypair* printed on the paper note. To spend this money, the recipient simply uses the *private key* printed on the paper note for signing transactions originating from this wallet address.

While *paper wallets* and *hardware wallets* are, currently, the most secure means of storing cryptocurrency, there are still several potential security issues concerning these devices. Both *hardware wallets* and *paper wallets* are affected should weak random number generation occur. If the parameters used for deriving the *private* wallet key are *predictable*, then deriving the *private* wallet key from its *public* address becomes computationally feasible.

For both types of wallets, it *must* be ensured that the *creation process* is *secure*. For *paper wallets*, if the *private key* is printed to a network printer on a compromised network, the cryptocurrency funds would then be available to whomever is able to sniff traffic on this network. For *hardware wallets*, an adversary might attempt to compromise firmware repositories so that *predictable* private keys are generated.

3.5 | Platforms and programming languages

Since the beginning of the *cryptocurrency* movement with *Bitcoin*, there has been great interest in using *blockchain* based solutions for implementing services. The most popular example of this idea is the *Ethereum*[12] smart-contract based cryptocurrency. In *Ethereum*, *smart contracts* are implemented as programs written in their language which they refer to as *Solidity*. *Solidity*[13] is designed around the syntax of *ECMAScript* but is statically typed. *Solidity* programs are compiled to a form of *bytecode* that is to be executed on an *Ethereum Virtual Machine (EVM)*. This *EVM* is a stack-based virtual machine that executes on all peers participating in the *Ethereum* network. Ethereum provides a web-based integrated development environment (IDE), known as Remix[13] for *Solidity* programming and *EVM* bytecode generation. As only the *EVM* code is distributed on the blockchain, and not the *Solidity* code, there is no *strict* requirement to use *Solidity*.

When developing applications that incorporate some form of *blockchain* based solution it is also possible and highly likely that there will still be some centralized component. For this reason, many cloud computing providers such as *Amazon AWS* and *Microsoft Azure* are now offering *Blockchain as a Service (BaaS)*. For example, just as one could rent a PHP server through a cloud computing provider, we are seeing an increasing amount of *private blockchain* services being offered by cloud computing providers. Instead of requiring an application developer to create the communication channels between the APIs for the various services used by the application, cloud computing providers are instead providing this infrastructure as part of their cloud computing service thus allowing an application developer to spend more time on the business logic of their application.

Just as *off-chain* transactions (eg, lightning network) is a proposed solution for blockchain scalability when working with monetary transactions, *off-chain* computation is a proposed solution for blockchain scalability when working with computationally intensive smart-contracts. On *public*, *permissionless*, smart-contract blockchains such as Ethereum, all peers arrive at a consensus on the result of a smart-contract as each and every peer executes each and every smart-contract. In order to prevent a smart-contract from consuming too much computational resource, Ethereum imposes a *gas limit* thus limiting the amount of instruction operations that the smart-contract can perform. In order to allow for more computationally intensive smart-contracts, all peers would need to come to a consensus on the result of the smart-contract execution *without* every peer having to execute the smart-contract. This is an ongoing research problem for which potential solutions[10] include; *trusting* smart-contract execution to *selected* peers, and, employing *proof-of-stake* voting to determine the validity of a proposed execution result of a smart-contract.

3.6 | Privacy and anonymity

Given that all activity which occurs on a public blockchain such as the *Bitcoin P2P network* is publicly viewable there have been many concerns raised considering user privacy. Excluding illegitimate activity, many users object to their transaction history being public for fear of user profiling or targeted advertising. As a result of these privacy concerns, research has been conducted into adding anonymity protections to blockchain systems.

One such example of adding anonymity to a blockchain is the *CoinJoin* method.[14] The *CoinJoin* method is an attractive means of anonymizing blockchain transactions as it requires no modification to the *Bitcoin* protocol. The only requirement for a *Bitcoin* transaction to be valid is that the amount of currency leaving a transaction (outputs) must be equal to the amount of currency entering a transaction (inputs). In keeping with this requirement, the *CoinJoin* method accepts sets of inputs and outputs from users seeking anonymity. The *CoinJoin* server then distributes the unsigned transaction to all parties in the mixing pool and requests their signature. The end result is a transaction of N inputs flowing to M outputs signed by all N inputs. Thus *CoinJoin* creates a *mixnet* with no direct link between any given input to the mixing pool and any given output to the mixing pool.

Unlike the *CoinJoin* method which works on the *Bitcoin* network without any modification to the *Bitcoin* protocol, *Monero*[15] is a *blockchain* based cryptocurrency designed from the ground up to protect user privacy. The core component of the *Monero* design is the use of *ring signatures*. A *ring signature* is a cryptographic signature which affirms that a message has been signed by one member of a group without revealing which group member actually produced the signature. Furthermore, *ring signatures* do not require the participation of the non-signing parties in the signature group. Therefore, using ring signatures, one can send money in *Monero* and it will only appear as originating from a group and not the specific individual who created the transaction.

The common design pattern between *CoinJoin* and *Monero* is the use of *mixnets*—that is, for N inputs, choosing only *one* true input and making the remaining $N-1$ inputs to be *decoys*. Under an *ideal mixnet*, the probability that an input caused an output is $1/N$. However, as *CoinJoin* and *Monero* both employ *mixnets*, they are also subject to the security issues associated with *mixnets*. Specifically, when *decoy* inputs are selected, their statistical properties should match those of the *true* input so that they cannot be detected as being decoys.[16] Users of *cryptocurrency mixnets* should also avoid selecting *publicly de-anonymized* transaction outputs as *decoy* inputs, as, given that their source is known, it becomes likely that the only *mixnet input* with an unknown source, is the one which caused the *mixnet output*.

4 | CONCLUSION

In this paper, we have provided a summary of the theory used to build distributed ledgers. We have shown how building this distributed ledger, or *blockchain*, has enabled software developers to create digital currencies which operate without the need of central financial institutions. We have also shown how this *distributed consensus* method can be used to enforce other rules on *digital assets*. In addition, we have summarized the current work and proposed solutions to address various issues surrounding *blockchain* technology.

ORCID

Qusay H. Mahmoud https://orcid.org/0000-0003-0472-5757

REFERENCES

1. McGraw L, Helbing C, Brodersen C. *Blockchain Technology: Preparing for Change Report*. Dublin, Ireland: Accenture; https://www.accenture.com/pl-en/~/media/Accenture/next-gen/top-ten-challenges/challenge4/pdfs/Accenture-2016-Top-10-Challenges-04-Blockchain-Technology.pdf. 2015. Accessed October 18, 2018.
2. Lamport L, Shostak R, Pease M. The Byzantine Generals Problem. *ACM Trans Program Lang Syst*. 1982;*4*(3):382-401.
3. Miles C, Harry K, Matthew WS, Arvind N. *On the instability of bitcoin without the block reward*. *CCS '16*. New York, NY: ACM; 2016:154-167.
4. Androulaki E, Barger A, Bortnikov V, et al. *Hyperledger fabric: a distributed operating system for permissioned blockchains*. *EuroSys '18*. New York, NY: ACM; 2018:30:1-30:15.
5. Schwartz D, Youngs N, Britto A. *The Ripple Protocol Consensus Algorithm Whitepaper*. San Francisco, CA: Ripple Labs, Inc. https://ripple.com/files/ripple_consensus_whitepaper.pdf. 2014. Accessed January 18, 2019.
6. King S, Nadal S. *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake White Paper*. Amsterdam, The Netherlands: Stichting Peercoin Foundation; https://decred.org/research/king2012.pdf. 2012. Accessed January 18, 2019.
7. King S. *Primecoin: Cryptocurrency with Prime Number Proof-of-Work White Paper*. Hong Kong: Primecoin Foundation; http://primecoin.io/bin/primecoin-paper.pdf. 2013. Accessed October 18, 2018.
8. Statista. Size of the Bitcoin blockchain from 2010 to 2018, by quarter (in megabytes) Web Page; 2018.
9. Bonneau J, Miller A, Clark J, et al. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. *Proceedings of the 2015 IEEE Symposium on Security and Privacy*. 104-121. IEEE; 2015.
10. Eberhardt J, Tai S. On or off the blockchain? Insights on off-chaining computation and data. Proceedings of the 6th European Conference on Service-Oriented and Cloud Computing (ESOCC); 3-15. Springer; 2017.
11. Popev S. The Tangle Whitepaper; http://www.tangleblog.com/wp-content/uploads/2016/11/IOTA_Whitepaper.pdf. 2018. Accessed January 18, 2019.
12. Wood G. Ethereum: A Secure Decentralised Generalised Transaction Ledger White Paper; https://gavwood.com/paper.pdf. Accessed October 18, 2018.
13. Ethereum. Solidity—Solidity 0.4.25 Documentation Online Documentation.
14. Bünz B, Bootle J, Boneh D, Poelstra A, Wuille P, Maxwell G. *Bulletproofs: short proofs for confidential transactions and more*. *2018 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA: IEEE; 2017:315-334.
15. Adam Mackenzie Monero Core Team. Improving Obfuscation in the CryptoNote Protocol Internal Investigation Report; 2015.
16. Möser M, Soska K, Heilman E, et al. An empirical analysis of traceability in the monero blockchain. *Proc Privacy Enhanc Technol*. 2018;*2018*(3):143-163.

How to cite this article: Mahmoud QH, Lescisin M, AlTaei M. Research challenges and opportunities in blockchain and cryptocurrencies. *Internet Technology Letters* 2019;2:e93. https://doi.org/10.1002/itl2.93