EUROPEAN UNION AGENCY
FOR CYBERSECURITY

CRYPTO ASSETS

An Introduction to Digital Currencies and Distributed Ledger Technologies

FEBRUARY 2021

ABOUT ENISA

The European Union Agency for Cybersecurity, ENISA, is the Union's agency dedicated to achieving a high common level of cybersecurity across Europe. Established in 2004 and strengthened by the EU Cybersecurity Act, the European Union Agency for Cybersecurity contributes to EU cyber policy, enhances the trustworthiness of ICT products, services and processes with cybersecurity certification schemes, cooperates with Member States and EU bodies, and helps Europe prepare for the cyber challenges of tomorrow. Through knowledge sharing, capacity building and awareness raising, the Agency works together with its key stakeholders to strengthen trust in the connected economy, to boost resilience of the Union's infrastructure, and, ultimately, to keep Europe's society and citizens digitally secure. For more information, visit www.enisa.europa.eu.

CONTACT

For contacting the authors please use evangelos.rekleitis@enisa.europa.eu
For media enquiries about this paper, please use press@enisa.europa.eu

AUTHOR

Nigel Smart, Thynnus Limited

EDITORS

Evangelos Rekleitis, ENISA
Angeliki Aktypi, ENISA
Athanasios Vasileios Grammatopoulos, ENISA

LEGAL NOTICE

COPYRIGHT NOTICE

EXECUTIVE SUMMARY

The European Commission on the 24th September 2020 adopted a comprehensive package of legislative proposals for the regulation of crypto-assets, updating relevant financial market rules [1], and is moving forward with creating a Pan-European blockchain regulatory sandbox facility to test innovative solutions and identify obstacles that arise in using Distributed Ledger Technnologies (DLTs) in the trading and post trading of securities. Crypto-assets may qualify as "financial instruments", in which case they fall under the Markets in Financial Instruments Directive (e.g.: tokenised equities or tokenised bonds). But there are also types that do not qualify as "financial instruments", such as utility tokens or payment tokens, generally refered to as digital currencies. Further, digital currecies when based on DLTs, like the Blockchain, are usually called cryptocurrencies; as opposed to centralized digital currencies.

These timely policy intiatives make evident that crypto-assets are a playground of not only technical, but also financial innovation that demands scrutiny in all its aspects. With this first introductory study focusing on the rise of cryptocurrencies & DLT, the European Union Agency for CyberSecurity is launching a series of information security studies in the area of crypto-assets to support policy-makers and raise awareness on the arising security and data protection.

The creation of BitCoin by Nakamoto [Nak08] in 2008 created a flurry of interest in so-called 'digital currencies'. The basic ideas of a blockchain, a consensus mechanism, and operations on a public ledger have potentially wide application outside of the narrow confines of creating a digital currency.

The technological ideas behind such distributed ledger technologies go back to way before 2008, often to the 1970s. What digital ledger technologies do is bring various technical components such as digital signatures, cryptographic hash functions, Merkle-Trees, consensus mechanisms, zero-knowledge proofs, secret sharing, together into an interesting combination which can address a number of application needs.

However, the hype behind such technologies understandably also creates unrealistic expectations as to what problems the technology can solve. This has led to a common quote of 'If you think your problem can be solved by blockchain, then you do not understand your problem'. This report aims to increase the understanding of blockchain technologies. It aims to explain the underlying technical concepts and how they relate to each other. The goal is to explain the components, and illustrate their use by pointing to deployed instances where the ideas are utilized.

[1] Digital Finance Package: Commission sets out new, ambitious approach to encourage responsible innovation to benefit consumers and businesses, https://ec.europa.eu/commission/presscorner/detail/en/IP_20_1684

CONTENTS

1 THE GENESIS OF BITCOIN

To understand how Bitcoin came about, and the resulting different forms of distributed ledger technology, one needs to see the genesis of the ideas as a continuum. Distributed Ledger Technologies (DLT) are at their heart simply a novel combination of existing simpler technologies. In this first section we aim to recap on these technologies and give their historical genesis.

1.1 HASH FUNCTIONS

The most important underlying cryptographic function in a DLT is a cryptographic hash function. These were introduced in the seminal paper of Diffie and Hellman [DH76] in 1976 as a means to construct digital signatures. A cryptographic hash function is a function with an (essentially) unlimited domain, but a fixed finite co-domain

$$H : \{0,1\}^* \longrightarrow \{0,1\}^h.$$

The length of bitstrings in the co-domain h is called the hash-length. Hash functions are used extensively in Computer Science, and generally they must 'look like' a random function. However secure cryptographic hash functions take this concept of looking random further and require three major security properties:

1. **The hash function should be pre-image resistant** – Given an output value y it is computationally infeasible to find a matching input value x such that $y = H(x)$. Thus it is said to be hard to solve the pre-image problem.
2. **The hash function should be collision resistant** – It is hard to find two values x and x' in the input's domain which map to the same output value, i.e. it is hard to find x and x' such that $H(x) = H(x')$. Thus it is said to be hard to find collisions in the hash function.
3. **The hash function should be second pre-image resistant** – Given an input/ouput pair (x, y) such that $y = H(x)$ it is hard to find another x', with $x' \neq x$, such that $H(x') = y$. Thus it is said to be hard to solve the second pre-image problem.

Modern secure cryptographic hash functions - such as the SHA-2 and SHA-3 families [NIS12] - are usually used with a hash-length that satisfy $h \geq 256$ to provide strong security.

1.2 MERKLE TREE

Invented in 1979 [Mer79], Merkle Trees enable a large amount of data to be hashed into a small key, whilst at the same time allowing a short proof that a given data item has been included within the final hash value. Consider the example in Figure 1.1, the value h_{00} is obtained by hashing m_{000} and m_{001}, i.e. $h_{00} = H(m_{000} \| m_{001})$. The value h_0 is obtained by hashing h_{00} and h_{01}, i.e. $h_0 = H(h_{00} \| h_{01})$. Whilst the final value h (known as Merkle Tree $root$) is obtained by hashing h_0 and h_1, i.e. $h = H(h_0 \| h_1)$.

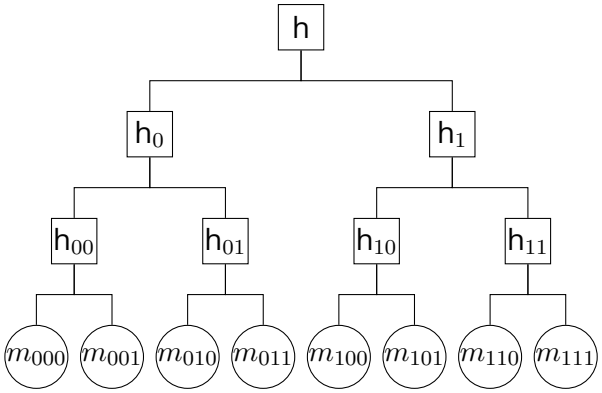

Figure 1.1: Merkle Tree of Depth $d = 4$

In this Merkle Tree we have stored the eight values $m_{000}, \ldots, m_{111}$, but we publish a single value h, which authenticates all eight values at once. To prove that a data item m_{100}, held at a leaf node, is authenticated by the root node h, the prover needs to reveal the path of hashes required to reproduce h given m_{100}. Thus to authenticate m_{100} in our example one reveals the values (m_{101}, h_{11}, h_0), since

$$h = H\,(\,h_0 \,\|\, H\,(\,H\,(\,m_{100} \,\|\, m_{101}\,)\,\|\, h_{11}\,)\,)$$

The key property is that a tree of depth d allows us to store 2^d values on the leaves, whilst requiring us to reveal only $d-1$ values to show that a data item has been included in the tree.

1.3 DIGITAL SIGNATURE

Digital signatures were also introduced in the famous paper of Diffie and Hellman [DH76]. They provide a means for a signer (who holds a private key sk) to authenticate a message m, so that a verifier (who only holds the public key pk) can verify that the signer did indeed 'sign' the message m with their corresponding private key. Formally we have three algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ such that

- KeyGen : Produces a public/private key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$.
- Sign : Generates a signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$.
- Verify : Verifies a signature on a message $\mathsf{Verify}(\mathsf{pk}, \sigma, m) \in \{0, 1\}$.

We require that valid sigantures verify, i.e.

$$\forall (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(), \quad \forall m, \quad \mathsf{Verify}(\mathsf{pk}, \mathsf{Sign}(\mathsf{sk}, m), m) = 1,$$

and that an adversary, who does not know sk, is unable to construct a valid signature σ, on a message m for the public key pk.

For the whole system to be secure, it has to be computational infeasible for an adversary who does not know the private key sk to generate a valid signature σ on a message m and a public key pk. Note that, as the name suggests, the private key sk should be kept secret by the signer while the public key pk can be publicly known by everyone.

Over the years many digital signature schemes have been proposed. The preferred signature scheme in deployed applications at the time of writing is the EC-DSA algorithm [NIS13], whose genesis goes back to the mid 1980's. The digital signature scheme used in Bitcoin is the EC-DSA algorithm specialised for use with the elliptic curve secp256k1 from [Sta10].

1.4 SECRET SHARING

A secret sharing scheme is a methodology to enable a secret, lets call this s, to be shared amongst a set of parties. This is done in such a way that if a certain set of parties come together (call a qualified set) then the secret can be recovered, but if an unqualified set come together then nothing can be learnt about the secret. The set of all qualified sets is called an *access structure*.

Various secret sharing schemes exist, and their constructions depends on the access structure. The simplest scheme is the full threshold scheme, which enables one to share a secret amongst n players in such a way that the only qualified set is the complete set of n players. To share a secret $x \in \mathbb{F}_p$ in this manner we write

$$s = s_1 + \ldots + s_n$$

and give s_i to player i.

The most famous non trivial secret sharing scheme is that of Shamir [Sha79] which supports so-called threshold access structures (n, t). This allows a set of n parties to share a secret s so that one requires $t + 1$ of the parties to come together so as to recover the secret. The construction is very simple, one selects a random polynomial $F(X)$ of degree t over $\mathbb{F}_p$ such that $F(0) = s$. Then party $i \in [1, \ldots, N]$ is given the value $s_i = F(i)$. The recovery of the secret by $t + 1$ parties is performed by Lagrange interpolation.

1.5 BYZANTINE AGREEMENT

The Byzantine Generals problem, or the problem of Byzantine Agreement, is a classic problem in distributed systems going back to 1982 [LSP82]. The basic problem is for a set of n entities to agree on a given value. If a proportion t of the entities are faulty (and actively so, a so-called Byzantine fault) then it is impossible to create consensus if $t \geq n/3$. A standard (classic) protocol by Lamport et al using digital signatures shows that agreement can be reached when $t < n/3$. See the book [CGR11] for a modern and readable introduction to various protocols in this space.

The problem was dubbed the Byzantine Generals problem by Lamport due to the analogy with a story of a group of generals trying to attack a city. If all the generals agree on attacking then the attack will succeed, if they all agree on retreating then they will not lose, however if there is disagreement then their army will be routed. The problem becomes complicated in the case where treacherous generals are hidden among the generals and they wish the army to be routed, and maybe alter, forge, or delete messages sent from one general to another.

1.6 ELECTRONIC CASH

The idea of creating a form of electronic money is older than the Internet. The most famous early example is a method of Chaum [Cha82] from 1982, which eventually led to the DigiCash project. Physical currency has an anonymity property that standard digital payment systems do not. A bank can trace who sent what to whom with digital payments, but one cannot trace bank notes as they pass through the economy.

The standard model of a financial payment involves four entities, the customer (who is paying for something), the merchant (who is receiving the payment), the issuer (the bank of the customer) and the acquirer (the bank of the merchant). In a traditional digital payment the payment is performed between the issuer and the acquirer, thus who paid what to whom and how much is visible to the banking

system. Cash transactions on the other hand happen between the customer and the merchant, with only the withdrawal and the final desposit being visible to the banking system. The precise link between the withdrawal and the deposit is being hidden.

Electronic Cash systems aim to provide the anonymity of physical cash in this regards. Chaum's system utilized blind signature schemes (a form of digital signature in which the signer does not know what they are signing) in order to create a form of digital cash. The DigiCash system still maintained the concept of a withdrawal and deposit from a 'bank', but the tracing of the withdrawn money through the system was anonymized.

A key problem with electronic cash, which DigiCash's complexity solved, is that of preventing double spending. With physical cash this prevented by being unable to copy a banknote or coin easily. In the electronic world copying bits is easy, thus avoiding double spending requires a different solution.

1.7 ZERO-KNOWLEDGE

One can consider a digital signature scheme as a proof that the signer knows some secret sk associated to a public key and a message. The interesting thing about the digital signature is that knowledge of the signature reveals no knowledge about the secret key, but the fact that it was used to produce the signature. This is an example of a more general concept called a zero-knowledge proof, which was introduced by Goldwasser et al in [GMR85].

Suppose you are given a statement S, along with a witness w that the statement is true. For example, this could be the statement that your bank account has more than 100 euros in it; with the witness w being the exact value of money in your account. A zero-knowledge proof is a means to prove to a third party that the statement is indeed true, without revealing any information about the witness. The security properties we require of a zero-knowledge proof are:

- **Correctness:** An honest prover can always convince a verifier that the statement is true if the prover holds a valid witness.
- **Soundness:** A dishonest prover finds it computationally hard to convince a verifier that a statement is true if the prover does not hold a valid witness.
- **Zero-Knowledge:** The proof itself reveals no information about the witness, but the fact that it exists and is known to the prover.

The advent of DLT technologies has created a huge new interest in Zero-Knowledge proof techniques, and their application, with many advances having been made over the last few years. We will discuss some of these below.

1.8 TIME STAMPS

The idea of using a 'blockchain', namely a sequence of hashed values, with each hashed value linked to the previous one by inclusion, goes back to 1990 and a method of time stamping of documents presented in [HS91]. See Figure 1.2 for an example. At a given time interval a series of documents, in this example called Tx0, . . . , Tx3 are hashed via a Merkle Tree into a single root node TxRoot1. This hash TxRoot1 along with a given timestamp value Timestmp1, a nonce Nonce1 and the hash of the previous block's hash PrecHash0 constructs one block of the chain that can be hashed to generate the PrevHash1. The PrevHash1 hash will be used to construct the next block on the chain, and so on, thus the PrevHash_i hashes are

used to link each block to the previous and next block. The sequence of linked blocks at the top of the diagram is called a blockchain.

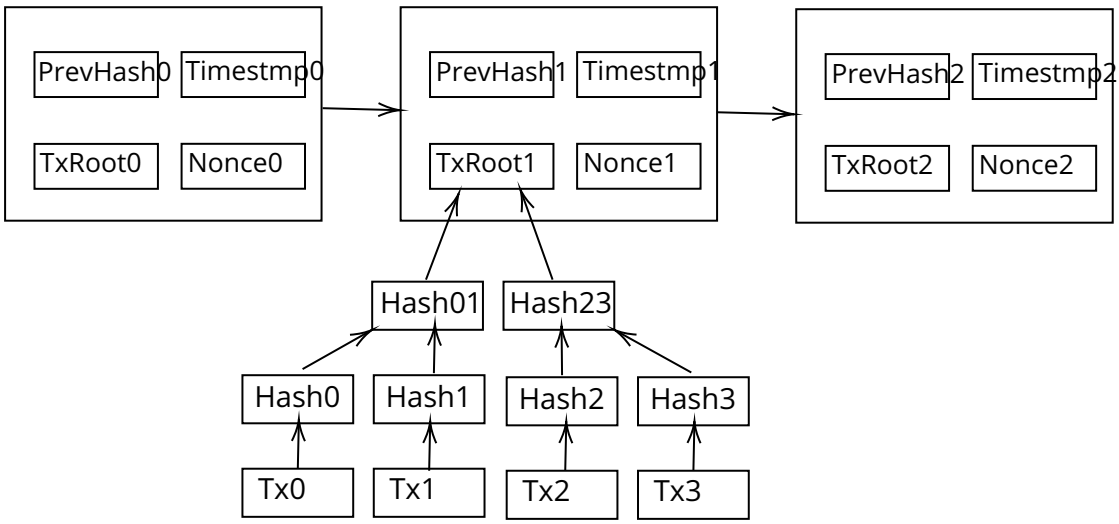

Figure 1.2: Time stamping documents using a blockchain and Merkle Trees

1.9 MULTI-PARTY COMPUTATION AND THRESHOLD CRYPTOGRAPHY

Multi-Party Computation (MPC) was developed in the 1980's in a series of works [Yao86, Yao86, BGW88, CCD88, RB89]. The basic concept of MPC is that it allows a set of mutually distrusting parties to compute an arbitrary function of their data. For example it allows two companies to compute a function on their respective customer databases, without revealing the customer details to each other. A closely related concept is that of Threshold Cryptography [Des94] in which the function to be computed is restricted to a cryptographic function on a secret key (for example the construction of a digital signature, or a decryption operation). In the latter case the secret is distributed amongst the parties using a form of secret sharing.

It was not until the last decade that the techniques for general MPC have advanced in order to enable useful calculations to be performed. The simpler case of Threshold Cryptography had earlier advances, but again for algorithms of interest (for example the EC-DSA signing algorithm using in Bitcoin) really efficient solutions have only become available in the last few years.

1.10 PROOF OF WORK

Proofs of work were introduced in 1993 by Dwork and Naor [DN93] as a mechanism to prevent denial-of-service attacks and spam email. The term itself was introduced by Jakobsson and Juels in 1999 [JJ99]. At their heart they provide a cryptographic proof that one party (the prover) has expended a given amount of computational effort, which is then checked by the verifier. This is performed by the verifier setting the prover a cryptographic puzzle. A cryptographic puzzle is a computational problem which is easy to generate and verify, but for which the solution is hard to compute.

The typical example of a cryptographic puzzle are hash-based puzzles. Here the puzzle generation is given by bit-string x, a difficulty parameter t and a hash function H with hashlengh $h \geq t$. The puzzle solver needs to find a value r such that the hash function H when evaluated on the bit-string $x\|r$ produces an output

whose last t bits are all equal to zero. On the assumption that the hash function is secure (both pre-image resistant and acts like a random function), the effort needed by the prover to solve this problem is expected to be 2^t hash function calls.

2 DISTRIBUTED LEDGER TECHNOLOGIES AND BITCOIN

We now discuss how to put the above components into a system, and what uses could such a system have. It is perhaps more illustrative to now approach the problem from the top, i.e. what problems are we trying to solve and then see how the above technologies can be put together into solving this problem. At the end we discuss the *specific* technology choices which the Bitcoin system makes. We use Bitcoin as an example as it is the most well known of all DLTs. We first pick apart the two words Distributed Ledger.

2.1 THE WORD 'LEDGER'

A ledger, according to Wikipedia, is *"the principal book or computer file for recording and totaling economic transactions measured in terms of a monetary unit of account by account type, with debits and credits in separate columns and a beginning monetary balance and ending monetary balance for each account."* In computer terms a ledger is nothing but a means to record data, thus in traditional banking systems a ledger might be implemented by a database. Such a database may (for example) record balances of individual accounts, but there is a subtle difference between such a database solution and a ledger. To change an account amount in a database one simply updates the record, whereas in the ledger we *append* the transaction to the ledger and this itself records the transfer of funds.

In particular a ledger only allows append operations, thus the state of the system can be rewound to see what it was in a previous time period. In a banking system, users get bank statements, which detail the indiviual transactions and these are usually kept for many years. Thus each transaction is appended to the datastore. Not only that but append is the *only* operation allowed. This means that prior transactions which have been added cannot be removed; in accounting terms they can be reversed by adding a counter transaction but never removed. Thus one can consider a ledger to be an *append only* database, thus the database is *immutable* (i.e. one cannot change events in the past). Hence the first question one must ask when considering DLT for a solution is

Question 1 *Does your application require data which is stored in an immutable form, i.e. would an* append only *database be suitable?*

The state of the ledger at time period i is assumed to be given by a state state_i. The transactions, or state transitions, to move from time period i to time period $i+1$, are given by $\text{Tx}_{i,j}$ for $j = 1, \ldots, n$. To obtain the new state state_{i+1} all the (consistent) transactions $\text{Tx}_{i,j}$ are applied (where consistency will depend on the application). In a blockchain system, such as in Figure 1.2, one records these state transitions by hashing the transactions together using a Merkle Tree, and then inserting the Merkle root into the blockchain as in Figure 1.2. Assuming you know the initial state of the system, and all the transactions, the current hash of the 'current block' (if accepted) will validate all the transactions, and hence the current state.

To avoid needing to rewind the entire system to the initial position one could also

store the current state and its hash within the current chain block; much like the value Timestmp is stored in the example in Figure 1.2.

2.2 THE WORD 'DISTRIBUTED'

Having picked apart the word ledger, and shown how this can be implemented using a blockchain, we now turn to the word 'distributed'. Again taking a definition from Wikipedia, a distributed system is a *"system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another"*.

Distributed databases already exist, for example a company's database may be cloned across different locations to enable security against accidental data loss, or to provide performance improvements by locating data close to where it is used. However, a major problem in distributed systems is to ensure coordination and trust between the different components. In the case of a distributed database one needs to ensure that read/write operations on one copy of the database result in the same operations when they are applied to a copy in another location. Much of the complexity of modern corporate database systems comes from the need to enable general read/write queries on the entire database in a way which ensures data consistency. Another reason one may want to apply distribution to data is that one does not trust a single data holder with the data. Thus we use distribution to protect not only against accidental, but also malicious, data loss or modification.

Distribution of data is thus an important property, but only some applications require it. Hence the second question one must ask when considering DLT for a solution is

Question 2 *Does your application require data to be stored in a distributed/decentralized manner for data loss prevention, security, or performance reasons?*

However distributed solutions, whilst giving many benefits, come with increased cost and complexity. The problem of ensuring consistency in the presence of network faults, be them accidential or malicious, is exactly the Byzantine Generals problem considered earlier.

2.3 THE WORDS 'DISTRIBUTED LEDGER TECHNOLOGY'

Thus a distributed ledger technology is at its heart an append only database in which the database is stored in a distributed manner. Then on top of this technology one can build applications, namely you decide what data is to be stored in the database and what a transaction to modify that database looks like. The combination of distrubution and ledger can have considerable advantages over using a traditional database solution.

The data storage technology can be simpler: Firstly, since we do not need to support all database transititions, only the appending of transactions. Secondly, in terms of obtaining consistency, we only need for the parties to agree on the transactions which needs to be added to move from one state to the next. Thirdly, as only hashes of both the state and the transitions are stored on the 'blockchain' the entire blockchain *could* be store publicly.

So once a decision for deploying a form of DLT has been made, the next question which arises is how one obtains the consensus. Consensus is provided by so-called *validators* (usually called *miners*) in the Bitcoin ecosystem. These validators actually perform two tasks, the first task is to verify that the state transitions being

applied to the blockchain are correct and consistent, and secondly to ensure that all parties agree on which transitions have been applied.

There are a number of ways to implement the validators, correspoinding to whether the users/deployers of the blockchain are considered trusted parties or not. The key distinction is between so-called provisioned blockchains and non-provisioned blockchains. At a high-level, provisioned blockchains are used when one has a natural trusted authority who can regulate and appoint the validators. In provisioned blockchains one uses traditional protocols to obtain consensus. In a non-provisioned blockchain there is no authority who is trusted to set up the validators. In this situation one requires different methods to obtain consensus based on so-called crypto-economics models.

2.4 PROVISIONED BLOCKCHAINS

In many commercial applications there is a natural set of semi-trusted validators. Consider a banking application within a given geographic region, the tier-one banks (clearing banks) within that region are naturally trusted. Whilst one may not trust an individual bank it is reasonable to assume that a majority will act correctly.

In a situation where one has a set of n parties in which one can trust that only $t < n/3$ of them will be motivated to act dishonestly one can deploy standard Byzantine Agreement protocols to validate the blockchain; i.e. to obtain consensus. Such protocols are highly efficient (compared to those used for non-provisioned DLTs), requiring only the passing of digital signatures between each party in a small number of rounds. We refer to [CGR11] for an explanation of some of these protocols in various situations.

This situation is called a *provisioned* blockchain as the set of semi-trusted validators provide the basic blockchain infrastructure for themselves (or others) to provide the applications upon them. A key aspect is that the validators for a provisioned blockchain are appointed or approved by a regulatory central authority, called the provisioning authority. The validators are only trusted (as a group) to provide the consensus mechanism, thus any data which is written to the blockchain must still be protected if the validator is not supposed to see the raw data.

2.5 NON-PROVISIONED BLOCKCHAINS

Sometimes, however, there are no natural semi-trusted parties to act as validators, or such a (relatively small) semi-trusted set is deemed to be a single point of failure. In such situations one moves into the realm of non-provisioned blockchains. In this scenario any one is assumed to be able to elect to become a validator, and security against corrupt validators is provided by having a large number of them. However, this comes at the expense of needing an alternative mechanism to provide consensus. There are two popular mechanisms for the consensus to be achieved; Proof of Work and Proof of Stake.

Proof of Work: In a Proof of Work system each validator competes to find a solution to a cryptographic puzzle, which depends on the next block to be added to the blockchain. For example the puzzle is to find a hash value terminating with a given number of zeros, where the input to the hash includes the hash of the next block. The first validator to solve the puzzle gets the reward of placing the next block onto the top of the blockchain, along with the associated solution to the puzzle. This solution is propogated through the validation network and then others try to place more blocks ontop. Due to propogation delays different validators

could have different views of the current top of the blockchain. However, as there is no point in validators trying to add blocks not at the current top, eventually a consensus will emerge as the blocks just below the top.

In Bitcoin this process is termed 'mining', and the assumption is that all blocks at a depth of six or below in the chain have been assumed to be permently committed to the chain, as the effort needed for a different set of six blocks to eventually become accepted is too great. The methodology works as long as the validators have an economic incentive to continue performing their task; we will return to this below.

As for a single miner the probability of solving the cryptographic puzzle before an other miner is relative small, and the cost of mining is very large it is common for miners to 'pool' their mining capabilities. Such mining pools search for a solution to the cryptographic puzzle, and when one member of the pool finds a solution the 'rewards' (see below) are distribued amongst the entire pool. Usually the pool ask the miners to solve an easier version of the initial problem – for example by reducing the difficulty t – and share the prise based on each miner's effort on the easier problem. By solving the easier problem, by luck a miner will eventually also solve the real problem.

The original Bitcoin paper [Nak08] argues that for an adversarial miner to have enough power to overcome the network, and place his own preferred blocks onto the top of the blockchain, he needs to control a majority of the mining network. This is the so-called 51% attack. However, in 2014 Eyal and Sirer [ES14] introduced the concept of selfish mining which showed that an adversarial miner can obtain some advantage when they get control of roughly $1/3$ of the network (which would be consistent with over Byzantine Agreement methodologies). See [NRS20] for a more recent analysis of selfish mining. At the time of writing the mining pool called F2Pool has about 16% of the mining capacity[1], with the top three mining pools having around $1/3$ of the capacity.

As Proof of Work requires spending computational power the process of mining/-validating becomes expensive, both in terms of electricity and heat dispersal. Professional mining is performed using application-specific integrated circuit (ASIC) and is dominated by a small number of 'big player' mining pools. It is also incredible damaging to the environment. An article from 2019 [SKG19] claims that Bitcoin mining accounts for 0.2% of all electricity consumption worldwide, and produces as much carbon dioxide as a metropolis the size of Kansas City (around 500,000 inhabitants). Other research from 2018 [KT18] calculated that Bitcoin mining produced, over a 30 month period, the equivalent of 3 to 13 million tonnes of carbon dioxide.

Proof of Stake: A Proof of Stake system aims to mitigate the environmental impact of Proof of Work systems. In such systems becoming a validator is open to anyone, thus one again does not adopt a central provisioning authority. The validators have some form of stored value (called their stake), often a certain amount of the underlying 'cryptocurrency'. At a given time instance the network decides, via a distributed coin-flip, who the next validator will be. The probability that a given validator is selected depends on various factors, for example it could be a uniformly random selection, or from a distribution which depends on the percieved wealth of the validator, or their percieved trustfullness.

Once the valdidator for the next block is selected, the validator creates a block of data from all parties wishing to add data, validates the data is correct and then adds it to the top of the chain (along with their digital signature to say they performed the operation). Other parties can check that the added block is correct,

[1] https://www.blockchain.com/pools

and if something wrong is found then the validator has some of their 'stake' removed, a process called 'slashing'. Thus validators who do not perform correctly are given a penalty. To avoid validators stopping certain parties from adding data to the blockchain, or giving preferential service to others, the validator for each block is randomly selected.

In both proof-of-work and proof-of-stake the correct behaviour of the validators is ensured by applying to the self-interest of the validators. Each validator wants to obtain the rewards available to those who contribute to validating a block, and no validator wants to be slashed. This appealing to selfish behaviour results in a form of economic incentive; with such modeling being dubbed crypto-economics.

Thus we are led to our third and fourth questions

Question 3 *Is your application of a DLT suitable for a provisioned or non-provisioned methodology for implementing the validators?*

Question 4 *If a non-provisioned blockchain, what methodology for consensus will be utilized?*

2.6 REWARDING VALIDATORS

All blockchains, whether provisioned or non-provisioned, rely on the validators to provide a service. A key issue which needs to be addressed, leading to a new field of 'crypto-economics' is how one can incentivize the validators to provide the service. Even in a provisioned blokchain validators have an incentive to 'free-load' off the work of others. One also has to protect the network against malicious validators doing incorrect actions as a form of denial of service attack.

There are various mechanisms to both reward and punish validators. We have already discussed, in Proof of Stake blockchains, how the penalty of slashing can be applied to a misbehaving validator. In addition if the selection of the next validator is done via a randomized method which is skewed by how 'trusted' a validator is to perform their task correctly, then validators who wish to validate blocks will not be chosen if they continually act in damaging the blockchain.

One way of rewarding validators used in almost all systems is that of 'transaction fees'. Thus entities wishing to place data onto the blockchain, or use its functionality, need to pay a transaction fee for the privilege. This transaction fee is then charged by the validators who perform the actual validation. This could be the set of validators who complete the Byzantine Agreement protocol in the case of provisioned blockchains, the validator who is chosen and successfully completes the validation in a Proof of Stake blockchain, or the validator who solves the cryptographic puzzle in the Proof of Work blockchain. In provisioned blockchains transaction fees are a natural methodology of rewarding validators and avoiding freeloaders. This can also be user as a method to prioritize transactions. Transactions with higher transactions fees are selected first for validation – as verifiers are trying to maximize their profit – and eventually they are served faster by the network.

A second methodology is to 'create' more digital currency whilst validating blocks. Thus the total amount of value in the system (measured in terms of units of the underlying 'currency') increases as more blocks are added to the blockchain. This assumes that the underlying digital currency will eventually be worth something in the real world, and thus the digital currency can in some sense be either spent on something useful to the validators, or can be converted into fiat currency.

Bitcoin adopts a combination of transaction fees and money generation into its reward mechanism for validators. It also has an inbuilt anti-inflationary money control mechanism, which means the total amount of Bitcoins which can ever be created by 'mining' is limited. Thus over time the validators will need to switch their main reward mechanism from mining new coins to transaction fees.

Thus we come to our next question for anyone wishing to deploy a DLT

Question 5 *Irrespective of your method of obtaining Byzantine Agreement, how will validators be rewarded (resp. penalised) for 'good' (resp. bad) behaviour?*

3 APPLICATIONS

As remarked above there is a natural fit between DLT applications and finance; a ledger is a standard mechanism for recording financial transactions and the reward mechanism needed for validators lends itself to some form of financial payment system.

BitCoin: The first system to deploy the above architecture for DLTs was obviously BitCoin. This aims to be a completely decentralized digital currency with no central issuing bank. The idea of BitCoin is to provide the benefits of cash, without the need to trust a govenment issuing authority. However, cash is a completely anonymous payment system: An external observer cannot work out how 'coins' or 'notes' pass through the system from one person to another. In addition an external observser cannot observe the sizes of payments being made, or even if a payment between two people has occured.

BitCoin only provides a limited form of anonymity. The identities of indivudals are hidden behind (hashes of) public keys, but the amount transferred between such public keys is completely in the clear. The transaction graph is completely public which means that analysis can be applied to the graph to determine how money flows through the system, thus limiting the level of anonymity actually provided [RS13, RS14]. One way of adding additional privacy into the BitCoin network is to use techniques such as zero-knowledge proofs (which we will come to later).

AltCoins: As is clear from the above discussion there can be many forms of digital currency; i.e. BitCoin is only an example (although the most famous). As we alluded to when discussing non-provisioned blockchains there is behind every such blockchain a form of digital currency. Thus each application *could* result in the introduction of a new form of digital currency. These 'alternatives' to BitCoin are collectively called AltCoins. The most famous of these are BitCoin and Ethereum, but there are well over one hundred such AltCoins.

The different AltCoins can be traded with each other on so-called digital exchanges, where they can also be converted into fiat currencies such as dollars or euros. The total value of each digital currency when expressed in terms of dollars is called its *market capitalization*. At the time of writing the top five coins (in terms of market capitalizations), their ticker symbols, the price per single coin and the total market capitilizations are:

Coin	Symbol	Price	Market Cap.
BitCoin	BTC	$18,115	$336,809,471,651
Ethereum	ETH	$551.47	$62,711,339,782
XRP	XRP	$0.62	$28,330,798,428
Tether	USDT	$1.00	$19,131,222,283
BitCoin Cash	BCH	$280.55	$5,226,206,168

Tether is slighly different from the other digital currencies as it is a so-called *stable coin*. The value of each Tether is always claimed to be one dollar. The idea is

that each Tether in circulation is actually backed up by one dollars worth of assets being held by the organizing company, Tether Limited. Although in 2019 the companies lawyer filed at the Supreme Court of New York[1] that one Tether only corresponded to $0.74.

Centrally Banked Digital Currencies: DLTs can also be used in traditional banking applications. Within each currency zone it is standard for the amount of money transferred between banks, to be recorded by the central bank on a system called RTGS (Real Time Gross Settlement). Thus when a customer transfers an amount from one account in one bank to another account in another bank there is a payment made between the issuing and the acquiring bank on the RTGS (actually many such payments are usually put together, leading to the 'delay' customers see when performing interbank transfers).

Such RTGS systems are a natural candidate for a provisioned DLT. The central bank acts as the provisioning authority, with the tier-one banks (a.k.a. clearing banks) providing validator nodes for the DLT network. Such a proposal, called RSCoin, was presented in [DM16]. A key problem here is that whilst this is a private blockchain the validators are competitors and thus one needs to ensure that the data on the chain placed by one bank is not readable by another. For example we do not wish bank C to see a payment made between banks A and B, as the size or occurance of such a payment is confidential information which should only be known to A, B, and the regulator. Again such problems can be solved via zero-knowledge proofs.

Financing and Securing Global Supply Chains: Trading of physical items can also be recorded, tracked, and paid for using DLT technology. Examples in this space include We-Trade[2] which provides a trade finance platform. The company has been set up as a joint venture between 12 European banks and IBM. Managing a global supply chain can also be relatively complex, and something which can utilize DLTs. To address this space IBM and Maersk have developed the TradeLens[3] platform to enable secure data exchange in the supply chain.

In a similar vein, but focused on the energy sector, a company called EnergyWeb[4] does a supply chain DLT in the electricity market. Their blockchain system brings together utilities, grid operators, as well as large energy consumers. In the oil and gas markets Chevron, ExxonMobil and others have teamed up to create a DLT solution called BlockChain For Energy[5].

For luxury and high value goods such as gemstones, paintings, fine wines etc. one can store the provenance and the ownership of items using a DLT; for example EverLedger[6] provides systems in this space. Of particular interest is the fact that one can store the 'atomic fingerprint' of a diamond on the blockchain in order to identify it and track its provenance.

Provenance not only applies to high value items; companies are also interested in determining the provenance of raw materials. For example in the food industry IBM, via its Food-Trust project[7], has been working with companies (supermarkets such as Walmart, producers such as Nestlé and Carrefour, as well as wholesalers) to ensure the provence of the food supply chain.

[1] https://www.scribd.com/document/408190972/Stuart-Hoegner-Affidavit-4-30
[2] https://we-trade.com/
[3] https://www.tradelens.com/
[4] https://www.energyweb.org/about/what-we-do/
[5] https://www.blockchainforenergy.net/
[6] https://www.everledger.io/
[7] https://www.ibm.com/nl-en/blockchain/solutions/food-trust

Provenance is also a key problem in the pharmaceutical sector. IBM, KPMG, Walmart, and Merck are using DLTs to attempt to secure the supply chain for drugs. This follows the introduction of the Drug Supply Chain Security Act in the United States. The companies have formed the Pharmaceutical Utility Network[8].

Other Applications: A companies most valuable asset is often said to be their staff. The large US based Workday company[9], which runs cloud based financial and human resource operations for companies, launched in 2019 a DLT to enable digital credentials in the HR (Human Resources) space. Enabling for example a health care professional to prove they have the requisite skill sets to different employers with a single trusted source of record.

A DLT provides a mechanism to record who owns what, and the transfer of such items between entities. Thus applications in finance can include the register of securities, a companies share register, ownership of bonds, valuable items and so on. Other applications can involve storage and transfer of KYC (Know Your Customer) data; for example institutions can share KYC data meaning an individual does not have to prove who they are to each institution they deal with in turn. There are various companies in this space, such as DTCC[10] and SETL[11].

The ability to store and record data, as well as provide a financial incentive via a digital coin, can allow interesting use-cases to be developed. For example, in Belgium one company Buck-E[12] uses DLT to provide a digital coin reward to children for cycling to school. This can also be extended to incentivize employees to cycle to work. Other applications include utilizing cryptocurrencies with computer game environments. In this space Forte[13] aims to develop a platform to make it easier to build games upon such a Distributed Ledger. Another company in this space is DapperLabs[14], probably most famous for its CryptoKitties application. Rarible[15] and Foundation[16] enable the storage and selling of so-called digital collectibles and digital art, so called NFT's (Non-Fungible Tokens).

The ability to transfer and store small amounts of 'value', has led a number of companies to investigate the usase of DLT in the music industry; as a way of distributing royalties to musicians. Examples here include Opus, Mediachain, Musiclife and eMusic[17].

[8] https://tinyurl.com/yycqjrhz
[9] https://www.workday.com/
[10] https://www.dtcc.com/blockchain
[11] https://setl.io
[12] https://buck-e.be
[13] https://www.forte.io/
[14] https://www.dapperlabs.com/
[15] https://rarible.com/
[16] https://foundation.app/
[17] https://tinyurl.com/yckl6sct

4 SMART CONTRACTS

At the core of many DLT applications, i.e. applications which run ontop of the base DLT technology, is something called a Smart Contract. Smart Contracts were initially made popular in by the Ethereum blockchain (which is considered the second largest cryptocurrency after BitCoin). A smart contract is a piece of computer code which will automatically execute 'on the blockchain' once certain conditions have been met.

For example, one could use a smart contract to act as a form of money escrow system: A user A pays some money to user B, but user B is not allowed to spend that money until a third user C gives their approval. If the approval is not provided within a given time window the initial payment from A to B is reversed. The point is that once entered into, the smart contract will be automatically executed by the validators. One application which was tested in this space in 2017 by Atlas Insurance from Malta and Aza in France was a travel insurance smart contract, which automatically paid out if a flight was cancelled or delayed[1].

Smart contracts can also be used to transfer data between entities securely under control of the data owner. See EncryptGen[2] for an example of this in the sharing of DNA data for medical applications.

[1] https://www.axa.com/en/magazine/axa-goes-blockchain-with-fizzy
[2] https://encrypgen.com/

5 SECURING SECRET KEYS

In a blockchain in which transactions signal ownership of some resource, for example ownership of the underlying cryptocurrency or title to some external object of worth (a valuable painting or a stock in a company) it is vital that the secret key used to sign the transactions which transfer ownership is kept secure. Loss of the secret key becomes equivalent with loss of the resource or asset. There have been many stories in the press in relation to key loss meaning people loosing access to millions of dollars worth of assets.

There are four distinct methodologies which can be used to secure the storage of such keys, all of which are deployed in various DLTs around the world. These distinct methodologies are not mutually exclusive. Indeed one finds applications which combine a number of the following methodologies together (for example by combining MPC and HSMs, or secret sharing and Multi-Sigs).

Hardware Security Module: A Hardware Security Module (HSM) is the traditional method to secure cryptographic keys used in large enterprises such as banks. These are special purpose computers dedicated to cryptographic operations which have gone through a rigourous certification procedure. They are hardened against tampering, and have strong control on how keys can be accessed which are stored within the HSM. They are most suitable for large corporate usage being relatively expensive, indeed most financial institutions will already have a large HSM footprint in house.

A major disadvantage of HSM technology is that it is less flexible than other solutions. Unlike software solutions HSMs are not 'elastic', meaning one needs to manage and purchase enough HSMs to cope with ones expected peak load. In addition for organizations with large amounts of HSMs, management becomes a big issue. Especially when one needs to deal with a heterogenous real estate, where each HSM supports different cryptographic algorithm choices due to the age of the equipement.

Multi-Sig: A Multi-Sig is a methodology which associated to each asset a set of public keys. Each associated private key is assigned to a given entity and an access structure is defined. When a transactions associated to this asset is desired to be executed, something akin to a Smart Contract is performed. The asset is only operated upon if enough digital signatures are collected which satisfy the access structure. For example one could have three distinct public keys and require at least two to authorize a given transaction. Thus if one key is compromised or lost, control is still maintained of the underlying asset.

The advantage of Multi-Sig's is that they utilize traditional digital signature technology; the signer needs to simply execute a traditional signing operation. The disadvantage is that processing the verification is more complex, i.e. one needs to be aware of the underlying access structure and wait for the requisite number of signatures to come in. Another disadvantage is that the access structure is public. This access structure may itself be a corporate secret; as it may reveal the internal approvals process for an operation which one may not want to leak to the world.

Secret Sharing: A classic way to store a secret so as to enable both recovery in case of one computer being lost/damaged and also to avoid theft of a key is to use secret sharing. In this method the key holder splits their key into shares n using a threshold secret sharing scheme (such as Shamir) with threshold t. The n different shares are then stored in n different places. The advantage then is that an attacker needs to break into $t+1$ places to recover the key, and the legitimate user is protected in that they can recover their key as long as they have access to $t+1$ of the storage locations.

A disadvantage of using secret sharing on its own is that to use the key the shares need to be brought back together into one location. This presents a single point of failure location, in that during key usage the secret is available to a potential attack.

Multi-Party Computation: One can address the issue of needing to bring the shares in a secret sharing based solution into one location to produce a signature by using MPC/Threshold Cryptography. Using this technology one can perform the signing operation without bringing the shares back together. This enables a normal digital signature to be produced, using whatever access structure is desired by the signer; and indeed the access structure can be dynamically obliviously to the verifier.

In some sense MPC solutions are the opposite of Multi-Sig solutions. In an MPC solution the signing operation is more complex, but the verification is as for a normal signature. In an MPC solution, unlike a Multi-Sig solution, one does not reveal the access structure needed to authorise a signature. The use of MPC and Threshold Cryptography in blockchain and other application areas has resulted in a NIST project to investigate the space for potential standardization[1].

[1] https://csrc.nist.gov/projects/threshold-cryptography

6 ZERO-KNOWLEDGE PROOFS

Whether recording data on a public blockchain, as in the case of BitCoin or Ethereum, or on a private blockchain, as in the case of an RTGS system, the data on the blockchain needs to be secured. Just as one encrypts databases in a standard database application, one needs to also examine how data is encrypted when stored on a DLT. The problem then is that the validators need to validate the data on the blockchain without seeing it. The validators may need to execute a smart contract, make a payment between two entities etc, and this needs to be checked to be correct. For example the validator may need to check that entity X has a certain value in their account before allowing that entity to spend it.

This need to provide proofs that certain statements are true without revealing the specific values which prove the statement to be true provides the massive interest in the blockchain community in zero-knowledge proofs. The applications have created an explosion of different Zero-Knowledge proof technologies. Which we will now summarize.

The statement to be proved is usually encoded as an arithmetic or binary circuit, or as a sequence of constraints in 'language' called R1CS. When below we talk about 'proof size' or 'time' this is always a function of the complexity of the function (i.e. the circuit size or the number of R1CS constraints). The reason for there being a multitude of different technologies is that the design space is quite large; with different applications requiring different trade-offs.

We outline the main tradeoffs below:

- **Proof size:** A Zero-Knowledge Proof will need to be stored on the blockchain, so the validators can validate it. Thus proof size is an issue if one is interested in minimizing the amount of data to be stored.
- **Prover Complexity:** The proof has to be generated by the prover, who holds the witness/secret data. Thus we may require that the prover can be run very efficiently.
- **Verifier Complexity:** The blockchain validators, and anyone else interested in the validity of a proof, will need to check the proof at some point. Indeed a proof is only ever created once, but could be verified many times. Thus the time needed to run the verification procedure could be important in many case.
- **Trusted Setup:** Some proof technologies require a form of trusted setup, namely the users of the Zero-Knowledge proof system need to trust some entity to set up some parameters at the very beginning.

As of today there are three different, but related, technoligies to provide the necessary Zero-Knowledge Proofs on a blockchain, these are SNARKS, BulletProofs and STARKS.

SNARK: A SNARK is a Succinct Non-Interactive Argument of Knowledge. They were introduced by Groth in 2010 [Gro10], and provide the best in class performance in relation to proof size and verifier complexity. They are reasonably fast for proof generation, but their main drawback is that they require a trusted setup

to be utilized within a system. The trusted setup is also dependent on the statement being proved.

BulletProofs: Introduced in two works, [BCC+16] and [BBB+18], of which the latter gave the name BulletProofs, BulletProofs are proofs whose main advantage is that they no longer require trusted setups which are statement specific. On the minus side their proof size is not as good as that achieved in SNARKs, and the prover and verifier time are also not that efficient.

STARK: A STARK is a Succinct Transparent Argument of Knowledge, and they were introduced by Eli-Ben Sasson et al in 2017 [BBHR19], with concurrent work in a similar vien leading to a system called Ligero [AHIV17]. They produce proofs which are bigger than both SNARKs and BulletProofs. They provide the best in class performance for the prover, but the verification is usually slower than that for SNARKs. Unlike SNARKs or BulletProofs they are require no trusted setup. Another benefit of STARKs is that their construction can be made secure against quantum computers.

6.1 APPLICATIONS OF ZERO-KNOWLEDGE PROOFS

It has already been remarked that BitCoin payments are not totally anonymous. Payment destination and source addresses are not protected, they are only 'tokenized' and the payment amounts are kept in the clear. One method to add true anonymity is to take a set of transactions and then pass them through a 'Mixer'. A Mixer is a technology used in many applications to provide anonymity, by breaking the link between the source of a message and the destination. It is used in some electronic voting protocols to break the link between a voter and their vote, or in messaging systems such as Tor to break the link between a sender and a receiver.

In the context of a cryptocurrency a mixer is implemented by having a single transaction which takes a *series* of inputs from a set of source addresses (or wallet identifiers) and then outputing the values to a series of destination addresses, whilst breaking the linkage between the transactions. This needs to be done in a way such that it is clear that it has been done correctly, without revealing the link. This is where Zero-Knowledge Proofs are used, to ensure the mix has been performed correctly. The classic example of this is the ZeroCash/Zcash protocol from [BCG+14], which uses SNARKS.

A BulletProof based solution called Zether [BAZB20] has been added to Ethereum, providing much the same anonymity requirements. An extension of Zether is used within JP Morgan Chase's JPM Coin implementation[1].

6.2 REMOVING TRUSTED SETUPS

As remarked above both SNARKs and BulletProofs require a form of trusted setup. As a simple example consider the following problem, a trusted setup might require the generation of two group elements P and Q on an elliptic curve for which the discrete logarithm is unknown between P and Q. An insecure way of creating P and Q would be for a trusted party to generate P at random and then create Q by setting $Q = [x]P$ for some random value of x known only to the trusted third party. However, this means the trusted third party knows the discrete logarithm x and could therefore mount attacks on the protocol[2]. A simple secure way of

[1] https://tinyurl.com/y3y2y3al

[2] Indeed this is the idea behind the famous backdoor in the NIST DUEL-EC-DRBG random number generator.

creating P and Q would be to hash a known string to the points P and Q using a trusted cryptographic hash-function. If the known string is published, then there is no way a party can learn the discrete logarithm (without being able to invert the hash hash function).

This simple example shows the problem with needing trusted setups. However for systems using SNARKs and BulletProofs, the trusted setup is more complex. However, we can perform this trusted setup using another form of cryptographic technology called Multi-Party Compuation (MPC). MPC allows a set of mutually distrusting parties to compute an arbitrary function on their private data. Using MPC one can create a so-called ceremony in which a set of parties performs the trusted setup, but in a manner in which no one party learns the 'secrets' behind the setup (the x in the simple example above). As long as the users of the system trust that at least one of the parties in the MPC ceremony is honest, then the users can trust the output of the trusted setup.

This strategy was most famously implemented in the setup for the Zcash system. The first such ceremony was performed in October 2016, with a second ceremony in August 2018 for the Sapling upgrade to the network. The underlying protocol is described in [BGG19].

7 CONCLUSIONS

This report inaugurates a series of security studies on the topic of Crypto Assets, Digital Currencies and Distributed Ledger Technologies. It presents the various aspects of the technology behind DLTs and their application.

From the prism of the creation of the Bitcoin, it discusses the various cryptographic primitives needed to provide a working system; viz. digital signatures, Merkle trees, hash functions, zero-knowledge proofs. Important topics addressed include how validators solve the consensus problem. What is the basic problem of Byzantine agreement and what approaches are available to solve this in both provisioned and non-provisioned block chains. It describes what a blockchain is and how an append only database maps onto standard financial systems such as ledgers, making blockchain a natural fit in financial applications. Moreover, it explains how data is placed on a block chain, the techniques that are used to validate state transitions on the blockchain and how Zero-Knowledge techniques can be used to solve these problems, while maintaining data secrecy.

Perhaps more importantly, this study poses 5 unanswered questions that decision makers and application designers need to consider before jumping on the blockchain bandwagon:

- **Q** Does your application require data which is stored in an immutable form, i.e. would an *append only* database be suitable?
- **Q** Does your application require data to be stored in a distributed/decentralized manner for data loss prevention, security, or performance reasons?
- **Q** Is your application of a DLT suitable for a provisioned or non-provisioned methodology for implementing the validators?
- **Q** If using a non-provisioned blockchain, what methodology for consensus will be utilized?
- **Q** Irrespective of your method of obtaining Byzantine Agreement, how will validators be rewarded (resp. penalised) for 'good' (resp. bad) behaviour?

No single answer can be provided, for each application and system will bear its own intricacies. But the inability to provide a clear answer to them, should indicate to decision makers that perhaps using a blockchain is not the best course of action.

BIBLIOGRAPHY

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. pages 2087–2104, 2017.

[BAZB20] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. pages 423–443, 2020.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. pages 315–334, 2018.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. pages 701–732, 2019.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. pages 327–357, 2016.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. pages 459–474, 2014.

[BGG19] Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. pages 64–77, 2019.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). pages 1–10, 1988.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). pages 11–19, 1988.

[CGR11] Christian Cachin, Rachid Guerraoui, and Luís E. T. Rodrigues. *Introduction to Reliable and Secure Distributed Programming (2. ed.)*. Springer, 2011.

[Cha82] David Chaum. Blind signatures for untraceable payments. pages 199–203, 1982.

[Des94] Yvo Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–457, July/August 1994.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. 22(6):644–654, 1976.

[DM16] George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. 2016.

[DN93] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. pages 139–147, 1993.

[ES14] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. pages 436–454, 2014.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). pages 291–304, 1985.

[Gro10] Jens Groth. Short non-interactive zero-knowledge proofs. pages 341–358, 2010.

[HS91] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. pages 437–455, 1991.

[JJ99] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In Bart Preneel, editor, *Secure Information Networks: Communications and Multimedia Security, IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS '99), September 20-21, 1999, Leuven, Belgium*, volume 152 of *IFIP Conference Proceedings*, pages 258–272. Kluwer, 1999.

[KT18] Max J. Krause and Thabet Tolaymat. Quantification of energy and carbon costs for mining cryptocurrencies. *Nature Sustainability*, 1:711–718, 2018.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[Mer79] Ralph Merkle. Method of providing digital signatures. US Patent US4309569A, 1979.

[Nak08] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Web document., 2008.

[NIS12] NIST Special Publication 180-4. Secure hash standard (SHS). National Institute of Standards and Technology, 2012.

[NIS13] NIST Special Publication 186-4. Digital signature standard (DSS). National Institute of Standards and Technology, 2013.

[NRS20] Kevin Alarcón Negy, Peter R. Rizun, and Emin Gün Sirer. Selfish mining re-examined. pages 61–78, 2020.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). pages 73–85, 1989.

[RS13] Dorit Ron and Adi Shamir. Quantitative analysis of the full Bitcoin transaction graph. pages 6–24, 2013.

[RS14] Dorit Ron and Adi Shamir. How did dread pirate roberts acquire and protect his bitcoin wealth? pages 3–15, 2014.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[SKG19] Christian Stoll, Lena Klaassen, and Ulrich Gallersdörfer. The carbon footprint of BitCoin. *Joule*, 3:1647–1661, 2019.

[Sta10] Standards for Efficient Cryptography Group (SECG). Sec 2: Recommended elliptic curve domain parameters. http://www.secg.org/sec2-v2.pdf, 2010. Version 2.0.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). pages 162–167, 1986.

ABOUT ENISA

The European Union Agency for Cybersecurity, ENISA, is the Union's agency dedicated to achieving a high common level of cybersecurity across Europe. Established in 2004 and strengthened by the EU Cybersecurity Act, the European Union Agency for Cybersecurity contributes to EU cyber policy, enhances the trustworthiness of ICT products, services and processes with cybersecurity certification schemes, cooperates with Member States and EU bodies, and helps Europe prepare for the cyber challenges of tomorrow. Through knowledge sharing, capacity building and awareness raising, the Agency works together with its key stakeholders to strengthen trust in the connected economy, to boost resilience of the Union's infrastructure, and, ultimately, to keep Europe's society and citizens digitally secure. More information about ENISA and its work can be found here: www.enisa.europa.eu.

ENISA
European Union Agency for Cybersecurity

Athens Office
1 Vasilissis Sofias Str
151 24 Marousi, Attiki, Greece

Heraklion office
95 Nikolaou Plastira
700 13 Vassilika Vouton, Heraklion, Greece

enisa.europa.eu