Blockchain and cybersecurity: a taxonomic approach

Stefano De Angelis, Gilberto Zanfino,
Leonardo Aniello, Federico Lombardi, Vladimiro Sassone

University of Southampton

October 2019

Contents

1 Introduction

Blockchain is a disruptive technology emerged in recent years to untold prominence. Firstly employed as a public ledger within Bitcoin's cryptocurrency [41], nowadays blockchain technology is considered as a paradigm for multi-party systems of organisations and enterprises. Besides the well-known, news-making, iconic applications to cryptocurrency, blockchain is breathing new life into subjects such as distributed and decentralised computing, as well as data assurance. In such fields, the impact of blockchain is showcased by its unmatched potential to transform '*work*' into '*trust*,' which can change the face of computing in untrusted environments.

Indeed, blockchain is a peer-to-peer distributed network of nodes that share a replicated data structure without a central authority. The shared data structure, also called the *ledger*, consists of chained blocks, each one linked to the previous through referencing its hash digest. Each block contains a list of records that witness transactions occurred among participants. Such transactions represent an exchange of assets between a network's nodes (e.g., exchange of crypto-coins in the Bitcoin blockchain). The process of block creation is called *mining*, and is carried out by special nodes of the network called *miners*, which periodically activate a distributed *consensus* algorithm to select new blocks to be added to the chain.

Once consensus is achieved, all the nodes update their local copy of the blockchain and agree on the same state.

The blockchain's linked data structure allows all the nodes in the network to have the same view of the ledger. This means that they agree on all the previous blocks and on their relative order, which leads to strong consistency of the information shared on the blockchain and its history. Indeed, even if only one record of a past block were modified or deleted on a single node, its hash digest would differ, and the following block would keep pointing to the original, unmodified block. In other words, in order to modify a block effectively, a node would need to also modify all subsequent blocks, each with the new hash the precedent block, till the end of the chain. Furthermore, for these changes to be reflected on other nodes' local copies of the blockchain, would require global agreement. Hence, unless all nodes agree on all such changes, the consistency of the ledger would be unmistakably compromised. This is indeed a crucial feature of blockchain, which ensures *integrity* and *non-repudiation* of the shared data, and therefore gives rise to a trusted system that does not rely on any centralised trusted third-party authority. Due to these security and data assurance guarantees, blockchain pushed its boundaries beyond the realm of cryptocurrencies and became a subject of study in several other fields, notably including supply chain, provenance, IoT and other sectors where multiple parties want to share a computing/storage infrastructure. Indeed, modern blockchains have been endowed with nearly Turing-powerful programming languages that can be used to implement, deploy and execute fully decentralised and tamper-resistant applications, called *smart contracts*.

Notwithstanding its promising advantages, blockchain raises issues of security, scalability and performance that limit its deployment in real systems. Blockchain-based systems being distributed over multiple nodes, they typically present a broad attack surface, as each one of the participants can come under attack. A crucial aspect of such distributed system is their chosen consensus algorithm, i.e., the protocol that in each blockchain the parties employ to agree on the occurrence and ordering of transactions. State-of-the-art consensus algorithms provide two sets of security properties, dubbed respectively *safety* and *liveness*. Consensus protocols differ on the properties they afford to users and the assumptions they make on the underlying network model. Several works have been proposed in the literature to evaluate such differences, as e.g. [9, 57, 60, 2], although a fair comparison is elusive due to several contrasting assumptions. Consensus is a paramount component of blockchain systems, since it strongly affects not just security, but also scalability and performance. As a matter of fact, there currently is no optimal approach, as each of the existing protocols offers a suboptimal tradeoff between all desired properties.

Against this backdrop, the study and understanding of the security of blockchain becomes paramount, both in terms of the properties afforded by the technology and in terms of its vulnerabilities. Although a number of studies exist on the latter, the former is a relatively unexplored area of research, which constitutes the focus of the present paper. In greater detail, our aim here is to study the relation between cybersecurity and blockchain, as well as to understand the opportunities and risks of adopting a specific blockchain system. Specifically, we evaluate *(i)* security issues of blockchain platforms and their consensus protocols; and *(ii)* whether blockchain is a technology that offers novel opportunities to tackle known, cutting-edge cybersecurity challenges.

We present a taxonomy of security properties relevant to blockchain, and analyse the extent to which various kinds of blockchain meet them. In order to answer this question in the most general context, we base our discussion on the crucial security properties underlying blockchain, viz., consensus. Specifically, our taxonomy builds on four types of consensus algorithms, namely *PoW* (proof-of-work), *PoS* (proof-of-stake), *PBFT* (practical Byzantine fault tolerance) and *PoA* (proof-of-authority). We believe that our four chosen protocols cover the overwhelming majority of existing blockchain systems and their properties [9, 18]. We leave the analysis of further algorithms to future work. In the present paper, we focus on the four major platforms that currently implement the above consensus protocols: *Bitcoin*, *Ethereum-PoS*, *Hyperledger Fabric* and *Ethereum-PoA*. Furthermore, we evaluate how these platforms can in turn offer additional security strengths or open new weaknesses. Finally, we propose five relevant security topics and we discuss their relation with blockchain and open challenges, namely *(i) data security*, *(ii) data privacy*, *(iii) trust*, *(iv) resilience* and *(v) forensics*.

Structure of the paper. This paper is organised as follows. In §2 we present our blockchain taxonomy and illustrate its main properties. In particular, §2.1 focusses on the distinction between *permissioned* and *permissionless* blockchain; §2.2 illustrates and exemplifies the main different ways adopted to establish consensus among nodes as to what are the blocks in the blockchain and their sequencing. The paper will contrast *P-of-X* algorithms against *Byzantine fault tolerance* algorithms. Following that, §2.3 will recall the fundamental notions of the main blockchain platforms, viz., Bitcoin, Ethereum-PoS, HyperLedger Fabric, and Ethereum-PoA. Our security analysis starts in §3 with the definition of the *security properties* to be considered in the rest of the paper. Following up on our work in §2, we naturally divide such properties into properties of the consensus protocols and properties of the platforms. In both cases, our properties are inspired by the standard CIA triad[1] classification of security properties in Cybersecurity. Having setting the science, §4 the provides the bulk of our *security analysis* in the form of two tables that describe the behaviour of each protocol and each platform of §2.2 and §2.3 with respect to the relevant security properties in §3. Finally, §5 focusses the discussion of five broad *security topics* for blockchain, each encompassing several of the security properties of §3: data security, data privacy, trust, resilience and forensics.

2 A taxonomy of blockchain consensus protocols and platforms

2.1 Public versus private blockchain

Blockchain systems can be classified on the basis of access rules under different permission models. Participants of a blockchain network have rights of: *(i)* accessing data on the blockchain (*Read*), *(ii)* submitting transactions (*Write*) and *(iii)* running a consensus protocol ad updating the state with new blocks (*Commit*) [31].

The works proposed by BitFury and Garzik in [5, 6], defines blockchain systems on the basis of these rights. Specifically, regarding Read operations a blockchain can essentially be divided in two classes:

- *public blockchain*: no restrictions applied on Read operations;
- *private blockchain*: a predefined list of entities is allowed to run Read operations.

The Write and Commit operations in turn identify other two classes of blockchain:

- *permissionless blockchain*: no restrictions on Write and Commit operations;
- *permissioned blockchain*: only a predefined list of entities is allowed to Write and Commit operations.

In other words, in permissionless blockchain, anyone can join the network and execute Commit and Write operations. On the opposite side, nodes of a permissioned blockchain are known at the outset, thanks to an authentication mechanisms, and only those authorised nodes can participate in Commit and Write on the blockchain network. Regardless of the identification process, a permissioned blockchain can be either public or private, according to whether only authorised nodes are able to execute Read operations. Table 1 below shows a comparison between the identified models.

Public permissionless blockchains, as e.g. Bitcoin, operate in hostile environments and require the deployment of crypto-techniques to coerce participants to behave honestly. These crypto-techniques

[1]Confidentiality, Integrity and Availability.

	read	**write**	**commit**
public permissionless	anyone	anyone	anyone
public permissioned	anyone	authorised participants	all or subset of authorised participants
private permissioned	restricted to a subset of authorised participants	authorised participants	all or subset of authorised participants

Table 1: Main types of blockchain systems

involve the usage of a cryptocurrency (e.g. ether on Ethereum) to reward participants, which can be stored on a digital *wallet*. Indeed, a cryptocurrency wallet stores the public and/or private keys for the accounts, and can be used to track ownership, receive or spend cryptocurrencies. Contrarily, *private permissioned* blockchains operate in environments where participants are authenticated. For this reason, permissioned blockchain can hold participants accountable for misbehaviour in ways that permissionless implementations cannot. Thanks to accountability, systematic violations can be detected over time and resolved optimistically. This is a substantial simplification, and permissioned systems benefits from fairness property that derive from it.

2.2 Blockchain consensus protocols

A core component of blockchain systems is the underling consensus protocol. It is employed to guarantee total agreement on the order of transactions and have decisive impact on the performance and security of a blockchain protocol.

Consensus in blockchain networks is achieved essentially following one of two different approaches:

- *lottery-based*, whereby a randomly elected leader proposes new blocks on the chain;
- *voting-based*, whereby a voting mechanism is carried out to elect a new leader.

These approaches target different blockchain models. In particular, lottery-based algorithms can scale to a large number of nodes, hence are more feasible in permissionless networks where the elected leader simply propose a new block by a broadcast to the rest of the network. However, in this protocols the election of a leader can be extremely expensive in terms of time and resources, leading to a potentially crippling performance degradation.

On the other hand, the voting-based approach is advantageous for permissioned networks, where the number of participants is limited and known. Voting consensus protocols afford lower latencies than the lottery-based: as soon as a majority of nodes agrees on a transaction, consensus is achieved. Yet, voting-based algorithms typically require intense message exchanges. Therefore, the higher the number of nodes in the network, the higher message exchanges is required to reach consensus. For this reason, in presence of large-scale networks, voting-based protocols performance may degrade due to intense communications, leading to very bad scalability.

Even though blockchain systems boast distributed consensus, the lottery-based approach differs from classical strong consistent consensus protocols, which implement total order broadcast and state machine replication. Indeed, blockchain lottery-based algorithms may admit multiple winners and, therefore, lead to forks in the blockchain. In this sense, such protocols can only guarantee a sort of *eventual consensus*, where the forks that potentially arise are eventually resolved in the future.

Eventual consensus is a fundamental concept in blockchain systems. It is often referred to as absence of *consensus finality*, whereby a valid transaction can never be removed from the blockchain once its block is appended to it [55, 56]. Instead, blockchains powered by the voting-based approach implement the classical strong consistent distributed consensus. This implies the use of classical *Byzantine fault tolerance* (BFT) protocols, which ensure low latencies and high performance as well as guaranteeing consensus finality to blockchain.

However, due to the inherent scalability limits of BFT, many hybrid protocols have been recently proposed by the blockchain community. These aim to boost the power and applicability of blockchain platforms by overcoming both the scalability issues of BFT protocols and the performance issues of lottery-based protocols. These algorithms are dubbed collectively as *Proof-of-X* (PoX). PoX protocols can be based on either a voting or a lottery approach, yet they can only guarantee eventual consensus.

In the rest of this section we describe two of the main lottery-based approaches, namely *proof-of-work* and *proof-of-stake*, the most common implementation of BFT, viz., the PBFT protocol, and finally the most prominent hybrid proof-of-X protocol for permissioned blockchain, namely *proof-of-authority*.

Proof-of-Work

The Bitcoin blockchain system is regulated by a new consensus protocol which ensures transaction order and prevents double-spending in a trust-less network: the PoW algorithm. This protocol consists of a computationally-intensive hashing task executed by the nodes. With PoW, the miners are in charge to do some computations to find a random number (also called *guess*) such that, if concatenated with the content of a block, 'hashes' to a number lower than a specifically-set target (see Fig. 1). Such target number is fixed according to the so-called blockchain *difficulty*, which regulates the average time spent by miners to solve the puzzle.

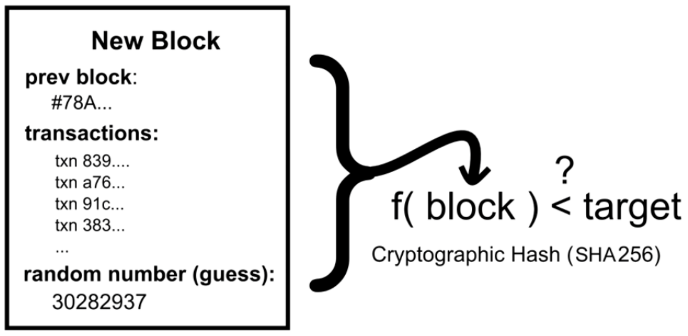


Figure 1: Proof-of-Work as a computational puzzle to solve a block

Once a miner solves a block, i.e., succeeds in the finding the 'guess,' that block is broadcast through the network for other nodes to accept. Once the block is accepted, all the correct nodes consider it as the latest in the blockchain, and start mining new blocks on top of it. For simplicity, we can say that once a miner creates a new block this becomes part of the chain. However, if multiple miners concurrently create ad propose new blocks, a transient *fork* is created. It this is the case, a fork is resolved over time because, by design, miners concatenate blocks on top of the longest chain. With the PoW, miners are incentivised to support the network honestly by a rewards mechanism. Namely, for every mined block, a miner receives a reward which is proportional to the fees each transaction include in its execution [41, 4].

When a node appends a new block to its local chain, it verifies its content and eventually accept it. Hence, once a node has validated and appended a block on the blockchain, the block becomes practically *non-repudiable* and *persistent*, unless an attacker can co-opt the majority of the miners' hash power. In this case, the adversary is able to create a chain fork or reverse a transaction. Assuming a majority of hash power controlled by honest miners, the probability of fork with depth n is $\mathcal{O}(2^{-n})$ [7]. Since the probability of forks decreases exponentially with their length, a suitable strategy for users to be reach a high degree of confidence that their transactions are permanently included in the blockchain (i.e., beating the double spending attack), is to wait for a small number of blocks to be appended to the one that contains them (actually, 6 blocks in Bitcoin). However, [24] shows that "*majority is not enough: Bitcoin mining is vulnerable*" if only 25% of the computing power is controlled by an adversary.

Although they provide strong integrity properties, PoW-based blockchains have a main drawback: *performance*. Their lack of performance is mainly due to the broadcasting latency of blocks on the network and to the time-intensive task of PoW. Indeed, thousands of miners spread all over the network are required to concur if PoW is to render tampering with transactions computational infeasible. To broadcast blocks over network of this size and topology can take very long. Thus, as a matter of fact, each transaction stored on a blockchain has a high confirmation time, which causes an extremely *low transaction throughput*. In Bitcoin, the average latency is 10 minutes, and the throughput is about 7 transactions per second, while for Ethereum the latency is on average 10 seconds with 24 transactions per second as throughput [7]. Moreover, PoW is *energetically inefficient*, leading a huge waste of money and resources to make the hashing computations.

Proof-of-Stake

Ethash is the PoW implementation of Ethereum. Like in Bitcoin, it suffers from the main drawbacks of probabilistic algorithms. Because of that, the Ethereum consortium proposed a consensus algorithm alternative for public blockchain to the classical PoW, namely PoS, which was first exemplified with the cryptocurrency Nxt [42]. PoS employs a deterministic (pseudo-random) protocol to select the next proposer. The PoS algorithm works as follows. The blockchain keeps track of a set of validators. Any node holding some Ethereum's cryptocurrency, the *ether*, can become a validator by sending a special type of transaction that locks up their ether into a deposit. The validators propose and vote on the next block, and the weight of each validator's vote depends on the size of its deposit (i.e., their stake). In the Ethereum's PoS implementation, called *Casper* [22], each validator's turn is determined by one of the following techniques:

1. *Chain-based PoS*: the algorithm pseudo-randomly selects a validator during each time slot (e.g., every period of 10 seconds might be a time slot) to propose a block. The block is then appended to the blockchain;

2. *BFT-style PoS*: validators are randomly assigned the right to propose blocks. However, the agreement on which blocks to add is done through a multi-round process, where each validator sends a 'vote' for a specific block. During each round and at the end of the process, all (honest and online) validators permanently agree on whether or not any given block is part of the chain. Note that blocks may still be chained together; the key here is that the consensus on a block is reached immediately, and its finality is not influenced by the length of the chain after it.

Differently from PoW, the PoS algorithms causes no waste of energy, because the algorithm demands no high computational effort. Importantly, a validator risks losing their deposit if the block they staked it on is rejected by the majority of validators. Conversely, validators earn a small reward, proportional to their deposit stake, for every proposed block that is accepted by the majority. Thus, PoS induces validators to act honestly and follow the consensus rules by a system of reward and punishment. As expressed by Vitalik Buterin, the creator of Ethereum, in a blogpost: *"in proof-of-stake, security comes not from burning energy, but rather security comes from putting up economic value-at-loss"* [8].

Despite the PoS security strengths, many blockchain based on PoS algorithms are being suggested which support a reviewed reward policy without penalties, as e.g. [47]. This leads to the *nothing-at-stake* problem, whereby validators are incentivised to sign blocks on any possible chain, as it does not cost them anything to do so. Worse still, in this scenario a validator is actually encouraged to act badly and create as many blocks as possible, opening the way to multiple forks and, as a consequence, to double spend attacks.

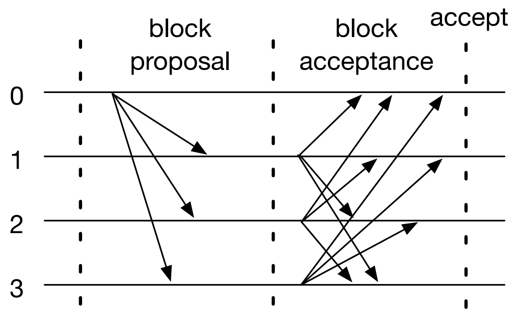
Practical Byzantine Fault Tolerance

In a Byzantine fault tolerance replication, where nodes may be subverted by an adversary acting maliciously, the most prominent consensus protocol is the so-called *PBFT* [10]. PBFT is an extension of the Paxos/VSR (*ViewStamped Replication*) [34, 43] family and it is characterised by a single-leader, view-change protocol. The algorithm proceeds in views, for each view there exists a leader and a set of replicas. Each view executes a *three-phase commit* protocol where replicas exchange messages to reach total order on transactions. In case of leader misbehaviour, all the correct replicas run a view change operation which starts a new view and elects a new leader. In an eventually-synchronous network, where messages are delayed and network partitions may happen but are eventually resolved, if an adversary controls f of the N network nodes, the PBFT consensus protocol guarantees strong consistency provided that $f < N/3$. It has been proved that in this scenario $N \geq 3f + 1$ nodes is a condition necessary and sufficient to guarantee Byzantine fault tolerance [10].

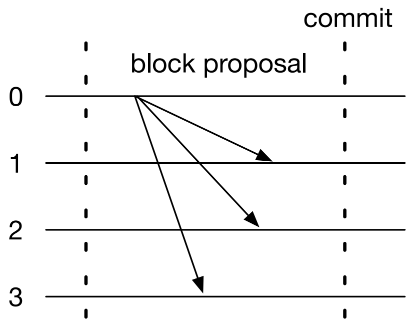
Proof-of-Authority

PoA is a family of consensus algorithms whose claim to fame is its increased performance over typical BFT algorithms; this indeed arises from lighter demands in terms of message exchanges. PoA was originally proposed as part of the Ethereum consortium for private networks and implemented with the protocols called *Aura* and *Clique* [46, 12].

PoA algorithms rely on a set of trusted miners called *authorities*. Each authority is identified by a unique *id*. Consensus in PoA algorithms relies on a *mining rotation* schema, a widely used approach to fairly distribute the responsibility of block creation among authorities [5, 26]. Time is divided into *steps*. In each step, an authority is elected as mining leader[2]. The way an authority is elected as leader differs in the two consensus protocols. Aura proposes a deterministic function based on UNIX times, which requires strong synchronisation assumptions on the network. Conversely, leaders in Clique are elected according to the height of the blockchain, and the elected leader is strictly related to the block number.



(a) Aura



(b) Clique

Figure 2: Example of message exchange steps for Aura and Clique PoA protocols. In this example there are 4 authorities with id 0, 1, 2 and 3. The leader of the step is authority 0.

[2]For the cognisant, as PoA has no PoW-like hash-based procedure, the consensus process is more appropriately called *minting*. However, for the sake of this presentation, we will continue using the wording mining.

As can be gathered from Fig. 2 above, these two PoA implementations work quite differently: both have a first round where the new block is proposed by the current leader, the so-called *block proposal*; then Aura requires a further round, called *block acceptance*, while Clique does not. A thorough comparison between Aura and Clique has been proposed by De Angelis et al. [18].

The PoA is a hybrid consensus protocol between the lottery-based and voting-based approaches, where leaders are elected according to a deterministic function. PoA protocols guarantee eventual consensus on transactions. Indeed, the lightweight leader election may lead to forks that eventually are resolved. Consequently, PoA cannot achieve instant finality but this is delayed in time. According to the concept of the longest chain in PoX protocols, a block in PoA is considered final when a majority of further blocks have been proposed, under the assumption that blocks are proposed at constant rate [46].

2.3 Blockchain platforms

In this section we describe the blockchain platforms considered in our analysis. We select two platforms belonging to permissionless blockchains typology, namely Bitcoin implementing PoW and Ethereum implementing PoS. As for permissioned blockchains, we pick Ethereum implementing PoA and Hyperledger Fabric implementing PBFT.

Bitcoin

Bitcoin [41] was the first popularised application of permissionless blockchain for a virtual currency. It is an electronic payment system based on cryptographic proof, which avoids the need of a centralised authority, e.g. bank. Normally, payment systems rely on trusted third-parties to process transactions, and such mediation is often subject to transaction costs. This limits the minimum practical transaction size and reduces the viability of small casual transactions.

The Bitcoin protocol focusses on the concept of *machine-to-machine* payment. The system relies on a flood propagated peer-to-peer network that processes transactions in a fully decentralised system. The order of transactions is guaranteed by an underlying 'lottery-based' consensus mechanism. To maintain a consistent, immutable, and therefore trustworthy, ledger of transactions, nodes share the Bitcoin *public ledger*, i.e., a blockchain with the list of all transactions ever made. Transactions are managed by asymmetrical cryptography as a chain of digital signatures. Each owner transfers value by digitally signing a hash of the previously received transactions with the public key of the new owner. In this way, only users who have previously received coins, can instigate new transactions. In Bitcoin, every node has a balance of the cryptocurrency it owns, identified by the history of its previous transactions. When a node receives a transaction, it verifies the sender's balance of crypto-currency by checking their previously used transactions. Hence, only transfers backed by existing funds can be accepted. Also, the Bitcoin protocol prevents '*double-spending*' of crypto-currency, by only validating transactions not previously used. Only unused transactions are accepted.

Ethereum public networks

Ethereum [23, 59] is the second main open source blockchain project, following Bitcoin. It builds on the idea of *smart contracts*, immutable programs deployed and executed autonomously on the blockchain. It is equipped with a(n almost) Turing-complete programming language which allows anyone to write distributed applications making use of blockchain data. This widens the application domain of blockchain, making it a '*programmable*' computational infrastructure with no centralised control.

Like in Bitcoin, the shared state is managed by 'enrolled' accounts, which feature a balance of the Ethereum's cryptocurrency, viz., the '*ether*.' The ether is the main internal crypto-fuel of Ethereum, and is used to pay transactions fees. Like in Bitcoin, nodes run PoW consensus to reach agreement on the order of transactions, and to maintain the consistency of the shared state.

There are two types of accounts: *externally owned accounts* (EOA), controlled through the use of asymmetric cryptography, and *contract accounts*, controlled solely by their smart contract code. The former are similar to standard Bitcoin accounts, whereas the latter can perform read and write operations on the blockchain through the action of smart contracts.

Ethereum transactions are activated by EOAs. They have two critical additional components over Bitcoin:

1. *gas price*: a scalar value representing the number of ether to be paid for each computational step of the transaction's execution;

2. *gas limit*: a scalar value representing the maximum amount of gas that should be used in executing the transaction.

Each instruction executed 'consumes' a given amount of 'gas.' Each transaction comes equipped with an overall limit to the gas it can consume while executing. This prevents runaway transactions, which accidentally or maliciously engage in never-ending computations. We leave to the reader to consider the crucial importance of this restriction for the mining process. Complementarily, gas price sets a fee system for computation, proportionally to its overall resource consumption. This includes not just bare computation, but also bandwidth and storage.

Ethereum introduces the concept of '*message*.' Messages are produced by contracts and essentially represent a call to a smart contract. These are virtual objects that only exist in the Ethereum execution environment, and are sent by contract accounts. A message contains:

- the sender of the message;

- the recipient of the message;

- the amount of ether to transfer alongside the message;

- the gas price.

Ethereum private networks

Currently, there are many implementations of the Ethereum protocol, most of them offer the possibility to be used in private permissioned settings. Two of the most interesting Ethereum clients are *Geth* [29], the Ethereum implementation in GOLANG language, and *Parity* [45], a RUST-based implementation. Both of them offer, through special configurations, the possibility of building permissioned private blockchain environments in which transactions are visible only to a subset of network participants. These Ethereum clients for private networks enable the integration of pluggable lightweight consensus algorithms.

These type of chains are mainly used in development *testnets*, where distributed applications are tested and debugged before deployment on the main Ethereum blockchain. Nevertheless, private networks are getting increasingly popular in the industry sector for business and enterprise use cases which require lightweight implementation, higher performance and higher privacy guarantees.

Hyperledger Fabric

Hyperledger Fabric [32] is a permissioned blockchain framework featuring a modular architecture in which each software component can be plugged, including the consensus algorithm. The distinguishing characteristic of Hyperledger Fabric is to offer an *authentication* and *authorisation* layer. Indeed, the system is operated by known entities, which can either be members of a consortium spanning multiple organisations or simply come from a single organisation. Each network participant is identified. They are enabled to issue transactions or to be involved in the consensus process only under proper permissions.

Hyperledger Fabric provides identification by leveraging on a Certificate Authority (CA) that issues a digital identity encapsulated in a X.509 digital certificate to each network participant. As the operating environment is more trusted, it allows to employ consensus schemas lighter than PoW, as for instance PBFT. This of course results in better performances.

Like Ethereum, Hyperledger is built on the idea of smart contracts running on the blockchain – *aka chain-codes* – able to update the ledger state appending transactions that take place among network participants. Assets can range from the tangible (e.g., real estate and products) to the intangible (e.g., contracts and intellectual property), and are represented in Hyperledger Fabric as a collection of *key-value* pairs.

In such a permissioned context, the risk of a participant intentionally introducing malicious code through a chain-code is diminished. All actions, whether submitting transactions, modifying the configuration of the network or deploying a smart contract, are first endorsed and then recorded on the blockchain only if deemed valid. Therefore, the malicious party can be easily identified and the incident handled in accordance with the terms of governance model.

Differently from other blockchain platforms, Hyperledger Fabric introduces the concept of '*channel*.' A channel represents a private blockchain *overlay*. In this context the overlaid blockchain can typically restrict access to authorised members of a consortium only. Basically, organisations on a Fabric network can establish a channel among the subset of participants who are granted visibility to a particular set of transactions. Each channel comprises a channel-specific ledger distributed across its nodes to store transactions occurred on it, immutably. Communications and transactions exchanged within a channel remain private and shared across only channel participants, enabling data isolation and confidentiality. The network of Hyperledger Fabric is composed by various actors with different roles:

- *organisation administrators*: in charge of deploying smart contracts on selected peers and releasing permissions to client applications;
- *client applications*: invoking smart contracts to perform read or write operations to a channel ledger, if holding proper permissions;
- *peers*: maintaining a ledger for each channel they are registered in; in addition, and if so designated, running smart contracts in order to query or update the channel-specific ledger;
- *orderers*: responsible for consensus, i.e., ordering the transactions which occur in all channels, packaging them in blocks and then distributing such blocks to nodes of appropriate channels.

Among these actors the orderers play a vital role, because they validate transactions and implement consensus, so realising the immutable storage of blocks on a channel ledger. Differently from Bitcoin and Ethereum-PoS, which rely on probabilistic consensus algorithms and may cause forks in the ledger, consensus in Fabric is carried out in through voting, and therefore avoids forks at the outset. In other words, any block generated by the ordering service is guaranteed to be final and correct. Thus any network participant has the same view of the accepted order of transactions.

3 A taxonomy of security properties for blockchain

In order to evaluate the security aspects of blockchain systems, in this section we produce a taxonomy of security properties, inspired by the classical tripartite definition of security as CIA: confidentiality, integrity and availability. Our proposed properties are organised in two categories: *(i)* security properties of the *consensus protocols* employed in blockchain systems; and *(ii)* security properties affecting the *platforms and technologies* underpinning blockchain services.

A blockchain can be described summarily as a *secure* distributed protocol which aims to regulate and store the transactions happening in a distributed network of parties. Such a system must satisfy the important properties of *safety* (or consistency) and *liveness*. Besides these, another important semantic metric in the evaluation of security and trustworthiness is the so called *fairness* property. Following seminal work by Francez [25], we distinguish two aspects of the fairness guarantees provided by a blockchain protocol: *validator fairness* and *client fairness*. The former relates to the consensus mechanism, while the latter is dictated by the architectural choices of the platform. We therefore start by formulating three security properties for blockchain consensus protocols as follows.

- *Safety*: the consensus algorithm should not do anything wrong during its normal execution; it prevents unwanted executions and ensures properties like consistency, validity and agreement. When an honest node accepts a transaction, then all the other honest nodes will make the same decision. If safety is violated at some point in time, it will never be satisfied again after that time.
- *Liveness*: something (good) eventually happens. This is a classical property in computing, aimed to prevent systems to meet safety requirements simply by staying idle and doing nothing. Liveness is indeed the dual property of safety: while the latter excludes the occurrence of some unwanted event ('something bad'), the former requires some wanted event ('something good') to happen. It is often used to ensure that algorithms and protocols keep making progress towards an end.
- *Validator fairness*: in a consensus algorithm, the leader election mechanism is fair if any honest node can be potentially selected to the set of nodes that will participate in the agreement to select the next block.

Turning to the security evaluation of blockchain platforms, we define six properties based on two fundamental concepts in the field of information security such as the *CIA triad* and user *profiling*, by which we mean a set properties which characterises user behaviour.

The whole blockchains act as trusted systems as long as the network is populated by a majority of honest nodes. A trustworthy system must be *dependable*, *resilient* and secure, ensuring properties introduced with the CIA triad such as confidentiality, integrity and availability [1]. In addition to the well known security and dependability properties, we suggest to focus on what we dub profiling properties, which are important for the evaluation of system fairness and trustworthiness. For profiling, we identified the relevance of the properties of *accountability* and *authorisation*. Such properties lead to fairness constraints which can be used to detect misbehaving participants. Our suggested security properties to evaluate blockchain platforms is as follows:

- *Confidentiality*: possibility for blockchain nodes to keep some of their transactions confidential; absence of unauthorised leaking of sensitive information owned by one or more nodes;
- *Integrity*: absence of improper alterations of the blockchain from unauthorised users;
- *Availability*: ability of the system to run correct services without interruptions;
- *Authorisation*: ability of the system to specify access rights and privileges to resources and to define permission roles to participants;
- *Accountability*: ability of the system to trace back the operations and the behaviour of a particular physical entity;
- *Client fairness*: willingness of the system to democratically accept transactions from any client without any preference.

4 Cybersecurity analysis

Blockchain can be represented by a multiple-layer architecture, composed of *(i)* network layer, *(ii)* consensus layer, and *(iii)* application layer. The first layer indicates the type of network where the blockchain is deployed in, the second states the consensus protocol employed, and the latter the technology and architectural choices implemented in the platform. Each layer is characterised by security flaws and inherits security features from the layers underneath. Application and consensus layers usually build on top of the network layer.

Verifying and establishing trust in the security semantics of a blockchain protocol is challenging. Broadly, a secure system should be resilient in the presence of adversarial conditions, and tolerate threats to its normal execution. In this section we propose a cybersecurity analysis of security aspects of blockchain platforms and their consensus protocols.

4.1 Evaluation of security properties

In this security analysis we assume a deployment of n nodes responsible for consensus, which communicate over an eventually synchronous network. The eventually synchrony model [20] simulates a real world network where messages can be arbitrarily delayed, but eventually are correctly delivered within a fixed and known time bound. We focus on the consensus and application layers and we evaluate respectively the consensus resilience and potential weaknesses introduced by our chosen blockchain platforms. Specifically, we present a comparison of the security properties listed in §3 of the blockchain consensus protocols *PoW, PoS, PoA* and *PBFT* and of the platforms *Bitcoin, Ethereum* and *Hyperledger Fabric.*[3]

	PoW	PoS	PBFT	Hybrid (PoA)
safety	eventual	eventual	yes	eventual
liveness	yes	yes	eventual	yes ⟨cf ⋄⟩
validator fairness	hw dependent	stake dependent	yes	yes

Table 2: Security evaluation of blockchain consensus protocols

Table 2 summarises the consensus resilience of the four targeted algorithms. We considered two algorithms for permissionless blockchain, namely PoW and PoS, and two algorithms for permissioned blockchain, PBFT and PoA.

The table shows whether the security properties of safety, liveness and validator fairness are met by the consensus protocols. Firstly, we observe that PoW and PoS, enjoy strong liveness. This property follows because the data is replicated locally on each node over large networks. Thus, safety is affected by the probabilistic nature of the leader election mechanism. Indeed, since these are lottery-based approaches, it may happen that two nodes are elected as leader at the same time. If then they each propose a new block, simultaneously, the result will be a fork in the system. In this case safety cannot be guaranteed due to inconsistent views of the blockchain. However, such forks will eventually be resolved, according to the specifics of the platform. For example, in Bitcoin a block is considered final when other six blocks are proposed after it, as the probability of a fork longer than that is vanishingly small [7]. For this reason, the safety of such protocols must be classified as *eventual.*

Another fundamental property of the security of consensus is validator fairness. In PoW, this property is strongly related to the hardware capabilities of a miner, because miners with a lot of computational power will have more chances to solve new blocks. On the other hand, the validator fairness of PoS

[3]In this analysis we do not consider the Aura-PoA consensus protocol since has been demonstrated in [18] to be unfeasible in eventually synchronous networks.

is strongly dependent on the '*stake*' owned by miners, i.e., richest miners will be advantaged in the proposal of new blocks.

Moving to permissioned blockchains, we observe they can rely on a higher level of trust than the permissionless ones, due to the presence of node authentication. The consensus protocols used in this context are either classical voting-based ones, such as PBFT, or hybrid, like the PoA. PBFT has been broadly demonstrated to guarantee safety in the eventually synchronous model, as long as there are $n \geq 3f + 1$ active nodes in the network, where f represents the number of potentially faulty nodes. Thus, as soon as there are $n \geq 2f + 1$ honest, non-faulty nodes, liveness is guaranteed [18]. On the other hand, in PoA, safety is only *eventually* guaranteed, because of the way PoA reaches consensus finality. Under the assumption of a majority of honest validators, liveness is instead guaranteed as long as at least one validator is active. In both protocols, validator fairness is guaranteed, since every node has the same chance of being elected as leader as the others ⟨⋄⟩.

	Bitcoin-PoW	Ethereum-PoS	Hyperledger-PBFT	Ethereum-PoA
confidentiality	no ⟨cf †⟩	no	channels	private txs
integrity	majority hashpower	majority stake	up to $3f + 1$	eventual
availability	yes	yes	up to $2f + 1$	yes ⟨cf ⋄⟩
accountability	n/a	n/a	yes	yes
authorisation	n/a	n/a	yes	yes
client fairness	no ⟨cf ‡⟩	no ⟨cf ‡⟩	yes	yes

Table 3: Security evaluation of blockchain platforms

We continue our analysis of blockchain security by turning our attention to platforms in Table 3. This table illustrates the security aspects of Bitcoin, Ethereum-PoS, Ethereum-PoA and Hyperledger Fabric. Both the permissionless and permissioned platforms inherit the safety (integrity) and liveness (availability) properties of their consensus algorithms. The integrity of both Bitcoin and Ethereum-PoS is strongly linked to where hash power and stake lie. Indeed, an attacker owing the majority of the hash power (or stake), could break the chain and maliciously inject a hard fork with subverted transactions [7]. In contrast, in Hyperledger and Ethereum-PoA, the integrity property is strongly tied to the safety property of their underlying consensus algorithms. In PBFT, it is guaranteed under the assumption of $n \geq 3f + 1$, while in PoA, it is only guaranteed 'eventually.' Despite strong availability, the full replication of the blockchain in the Bitcoin and Ethereum-PoS platforms leads to a lack of confidentiality, as each node in the network will have access to the entire ledger of transactions [33] ⟨†⟩. However, on these platforms it is possible, through the use of wallets, to offer some confidentiality by hiding the amount of cryptocurrency and the identity of the accounts one holds. Contrarily, confidentiality in both Hyperledger and Ethereum-PoA can be guaranteed through the use of channels and private transactions, respectively.

Hyperledger Fabric and Ethereum-PoA can enforce the so-called profiling properties of authorisation and accounting, as defined in §3. This is because nodes are authenticated. Authorisation is guaranteed by managing the permission of each node. Accountability is achieved by tracing the interaction of nodes with the blockchain [30]. This is not so in public permissionless blockchains like Bitcoin and Ethereum, where transactions are anonymous and users are not authenticated. This implies that actions are not attributable to specific entities [39, 33]. Hence, profiling properties are not available on these platforms.

Lastly, we evaluate the property of client fairness. Permissioned blockchain benefits from fairness guarantees in that each client's transactions are processed without any preference or priority. On the contrary, the execution of transactions in permissionless blockchain is costly (either hardware intense or stake intense), hence making incentive mechanisms for miners necessary. Bitcoin miners receive fees for processing transactions, while Ethereum-PoS transactions are embedded with a reward (*gas*). This means that low-rewards transaction may be stalled forever waiting to be processed [58]. Incentives mechanisms for permissionless blockchain, like Bitcoin and Ethereum-PoS, lead therefore to a lack of client fairness ⟨‡⟩.

It emerges from our analysis that permissioned blockchain as Hyperledger Fabric and Ethereum-PoA achieve better accountability, confidentiality and client fairness, and so enhance the system's fairness and trustworthiness. Conversely, permissionless settings can guarantee strong integrity and liveness properties as soon as the requirements on a majority of honest nodes are met.

This analysis will be used in the next sections for the evaluation of these blockchain platforms under the security services of §5, which we refer to as *security topics*.

5 Security topics

Building on the analysis of §4, here we reflect on interplay between blockchain and some prominent security topic. We identified five main ones:

- *data security*;
- *data privacy*;
- *trust*;
- *resilience*;
- *forensic*.

	data security	data privacy	trust	resilience	forensics
confidentiality	✓	✓	✓		
integrity	✓		✓	✓	✓
availability	✓		✓	✓	
accountability					✓
authorisation	✓	✓	✓		✓
client fairness			✓		

Table 4: Security topics and their relation with security properties.

Each of the topics we consider depends on one or more of the security properties treated in §3, as we illustrate in Table 4. In this section we describe current solutions and open challenges for each of the topics with respect to the different blockchain platforms presented in §2.3

5.1 Data security

Data security concerns the protection of digital data stored in a database from destructive forces and from the unwanted actions of unauthorised users, such as a cyberattack or a data breach [51]. Thus, according to our taxonomy, data security clearly impinges on the properties of confidentiality, integrity, availability and authorisation. A system designed to provide data security must therefore ensure those properties and employ:

- *encryption*;
- *authentication*;

- *replication.*

Specifically, encryption is used to store data so as to make it unreadable in case of a breach or intrusion. Encryption can be both performed in hardware as well as in software, and its security depends on the strength of the algorithm chosen and the length of the key used [52]. Authentication is used to avoid unauthorised access to other users' data. Modern authentication systems combine at least two-factor authentication and/or biometric technologies (when applicable) [38]. Replication provides fault-tolerance in case of data loss, unavailability or data tampering. Specifically, data can be maintained by *n* servers so that in case of a fault in one replica, there remains other *n-1* available [50] to use. Backup is indeed a form of replication that can be used instead to recover from compromised data.

Blockchain provides high availability since it can distributed on thousand of nodes, and can thus resist DDoS attacks. Because of its replication, it is considered too hard for an attacker to tamper with data in the blockchain. However, although it provides high availability and strong integrity compared to traditional databases, to employ a blockchain as a database remains today an open and challenging problem [7, 26]. Indeed, storing data on a blockchain means replicating data among all participants. Thus, to store encrypted data the user must endeavour to keep their private keys secret. Although the concept of '*wallet*' arose to keep such keys secure, it still leaves a big problem open: namely, if a wallet is hacked, the attacker obtains the key to decrypt data that is publicly available on the blockchain. This stands in stark contrast to what would happen in a database, where hacking an encrypted database requires the attacker to first break the database, then steal the decryption key and finally decrypt the content. It seems therefore quite obviously harder to steal data from a database rather than from a blockchain, given the further layer of security the database presents.

Furthermore, storing large amounts of data on a blockchain is another challenging aspect, because it means filling the physical mass memory of all user joining the platform with petabytes of data that they do not really need. Solutions have been proposed to adopt an off-chain database –which can be a normal database or a permissioned blockchain– to store the data, while keeping on-chain (i.e., on the public blockchain) just the hash of such data [26, 37].

5.2 Data privacy

Data privacy (or simply *privacy*) is a branch of information security concerned with managing, storing and sharing personal data with third parties. Privacy is an interdisciplinary topic not limited to tools and technologies, but also intertwined with laws and regulations such as GDPR, HIPAA, GLBA, or CCPA [36]. According to our taxonomy, data privacy required the properties of confidentiality and authorisation. Even though such properties are pivotal also in data security, the two concepts should not be confused. Indeed, a significant difference between security and privacy is that the former aims to protect data from compromise by external attackers and malicious insiders, while the latter aims to governs how data is collected, used and shared [40].

Privacy-enhancing technologies (PETs) are methods that allow users to protect their privacy [53]. Two families of PETs exist:

- *soft privacy technologies*;
- *hard privacy technologies.*

Soft privacy technologies assume that the third-party can be trusted to process data. The data protection model is then mainly based on compliance, consent, control and audit. Example technologies are *access control* and *tunnel encryption* (SSL/TLS) [19]. Hard privacy technologies assume that third-parties cannot be trusted and no single entity can be allowed to violate the privacy of the user. Thus, the data protection model relies on data minimisation to reduce the amount of personal data stored without losing the functionality of an information system; an example technology is the *onion routing* (TOR) [19].

A permissionless blockchain can be seen as an example of hard privacy technology. Indeed, Bitcoin can hide the identity of the users behind a pseudonym (i.e., the hash of a user public key) that nobody firmly can associate to an identity. This allows Bitcoin users to issue anonymous transaction of cryptocurrency. However, a different permissionless blockchain can intentionally decide not to provide such anonymity guarantee in the interest of providing accountability. We can then see anonymity and accountability as opposite properties. Ethereum, for instance, is based on the concept of 'EOA,' i.e., a digital identity. Similarly, on a permissioned blockchain each user has their pair of public/private key, thus all transactions can be traced back to the user, and we have accountability rather than anonymity.

Data protection in such technologies can be achieved by means of two main approaches:

- *data anonymisation*;
- *data masking* (or *obfuscation*).

Data anonymisation [54] aims to sanitise *explicit identifiers*, *quasi-identifiers* and *sensitive attributes* of a dataset in order to make it harder to trace the available information back to the data subject. Sanitisation operations are applied to the entire original dataset before it is shared with a third party. The main approaches are *k-anonymity*, *l-diversity*, *t-closeness* and *differential privacy*. However, all of these approaches relies on the dataset becoming available to the third party only after the anonymisation. This model cannot be ensured with blockchain, since the blockchain itself should contain the original (not anonymised) dataset.

Data masking [48] works by replacing the original data with modified content that look real and appear consistent. The main approaches are *substitution*, *shuffling* and *encryption*. Substitution and shuffling are similar, and aim to modify some entry of a record with other values. However, anyone with some background or side knowledge about the original data can then apply a 'What If' scenario to the data set, and attempt by inference to piece back together a real identity. Such approaches are therefore far from ideal for a blockchain. In turn, encryption requires that a key based on user access rights is applied to view the data. Although this can be a good solution, it introduces the same problem discussed above in the case of security: where and how to store the keys securely.

Thus, privacy is a big challenge for blockchain. Two directions to overcome such challenges worth of investigation are the integration of '*homomorphic encryption*' [27] and of '*secure multi-party computation*' (SMC, also referred to as *privacy-preserving computation*) [16]. Homomorphic encryption is a form of encryption that permits to compute on ciphertexts to extract an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext. Using such an idea and possessing the right keys, one can conduct data analysis and computation on encrypted content of the blockchain, and preserve privacy of both inputs and outputs. Similarly, SMC allows a set of computing entities to jointly compute a function over a set of data distribute among them, without any of them having to disclose their data to the others. which obviously can be used to implement privacy-preserving computation.

Despite the fact that it definitely constitutes a big challenge, using blockchain for privacy would unleash great potential through the application of *smart contracts* for privacy-preserving computation. Indeed, using smart contract we can obtain the additional bonus of tamper-resistant code, which might be used for the regulatory obligation tasks, like consent and notice.

5.3 Trust

Trust is a pivotal topic in cyber security. The internet before decentralised computing worked on the 'client-server' paradigm, which is the model of a 'trusted third party,' meaning that users are required to trust the entities they interact with. *Cloud computing* demands extreme trust in this sense, as users elect to completely delegate control and storage of their data, as well as the execution of their applications, to the cloud provider, in exchange for convenience and economy of scale. To wit, *distributed computing* rejected this centralised paradigm already several decades ago, on the grounds of efficiency and fault-tolerance. Blockchain brought *decentralised computing* to the internet centre stage, so that we now look at decentralisation also as a trust mechanism which allows us to exercise caution about online interactions with third parties. Indeed, switching the paradigm to "third parties cannot be trusted," we take on board that fact that internet entities can behave maliciously (cf. e.g. the Byzantine problem [35]).

According to our taxonomy, trust depends on confidentiality, integrity, availability, authorisation and client fairness. When we consider the three classic CIA properties, we observe that trust is strictly related to data security. Indeed, data security only make sense if we have a system we can trust sufficiently. To exemplify this, suppose we have built a system that employs all main approaches to data security perfectly, i.e., encryption, replication and authentication. If the user cannot trust that this system is adequately protected from attackers (external or internal) and will not be 'owned' by a malicious entity, then all the security mechanisms are for nothing. Since in the real world a cyber attack is detected on average 197 days after succeeding [49], the user has to accept that their data are potentially compromised the moment they are put on the system.

Trust is therefore crucial to ensure effective data security and privacy. So far, blockchain is the unique software-based alternative to the 'trusted third party' model. In blockchain trust is achieved by distributing it and building it through proof-of-work and other consensus mechanisms, in order to tolerate up to a minority of malicious nodes. All alternative solutions are based on trusted hardware. Specifically, with '*trusted computing*' (TC) the machine will consistently behave as expected, and such behaviour is achieved by loading the hardware with a unique encryption key inaccessible to the rest of the system. Systems like Intel SGX [15] are an example of TC.

Along TC, the Hyperledger consortium proposed *Sawtooth* [44], a blockchain employing a consensus algorithm called *proof-of-elapsed-time* (PoET) [11], that is not Byzantine fault tolerant but promises high performances. To enforce the security of the platform, they integrate the Intel SGX enclave functions within the validator of the PoET consensus. In that way, the task of generating trust is moved from software to hardware.

5.4 Resilience

Resilience is a topic orthogonal to trust. It describes the ability of a system to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation. Specifically, blockchain introduces a novel problem that is not found in other technologies, i.e., how to modify the underlying protocols. In fact, due to its integrity guarantees, blockchain does not easily tolerate modifications to its underlying protocols and the algorithms. Imagine for instance trying to change the underlying hashing algorithm. Now, there is something that would invalidate the whole chain! However, sooner or later any system will need to be updated, and so will blockchain-based ones. Returning for instance to the hash function that links blocks to each other, it is quite possible that it will eventually have to be updated to, for instance, a quantum-resistant hash functions. Yet, changing the hash function will invalidate all the blocks as well as the entire linked structure, which would require to recompute all the hashes linking each blocks to the blockchain. That is obviously infeasible. An alternative solution is to fork the blockchain after a protocol update. As the name suggests, a fork will cause the blockchain to split, so that two parallel blockchain will be spawned. There are two kind of forks:

- *soft fork*;
- *hard fork*.

A hard fork is when nodes of the newer algorithm no longer accept the older version of the blockchain, which creates a permanent divergence between the two versions. Adding new constraints to the code essentially creates a fork in the blockchain: one path follows the new, upgraded blockchain, and the other continues along the old path. Generally, after a short time, those on the old chain will realise that their version of the blockchain is outdated or irrelevant and quickly upgrade to the latest version. In other words, a hard fork requires all nodes to upgrade to the latest version of the blockchain client. A soft fork is instead backward-compatible. This kind of fork requires only a majority of the miners to upgrade for the new rules to be enforced.

According to our taxonomy, Resilience relates on integrity and availability. Indeed, after a fork some transactions can be rejected, as they do not follow the new protocols. This results in both availability and integrity to be violated, given that the fork temporarily creates two different blockchains that eventually will be resolved in a single branch, invalidating the other.

Some preliminary study of the quantum threat for blockchain has been conducted by Gheorghiu et al. [28], who observed that none of the existing post-quantum digital signature schemes proposed so far satisfies all the requirements of a distributed ledger system, namely small size and efficiency.

5.5 Forensics

Digital forensics is a branch of forensic science that regards the recovery and investigation of material found in digital devices, often in relation to suspect computer crimes. According to out taxonomy, forensics relies on integrity, accountability and authorisation. Blockchain can be used as a forensic platform because of its integrity guarantees. Indeed, in forensics it is crucial to trust to the authenticity of the data in case of investigation. It is also prominent to ensure accountability in order to be able to check the history of accesses and modification to specific element. Some blockchain platform may result anyway hard to use for accountability. Indeed, as we described earlier, Bitcoin hides the identity of the users. Some interesting solutions have been proposed as forensics toolkit to analyse the history of Bitcoin transactions and understand and trace payments in cases of ransomware [3].

6 Conclusion

Blockchain is gaining traction in the fields of distributed computing and data assurance to implement trustworthy and secure data-shared infrastructures. Although blockchain seems to provide outstanding advantages in cybersecurity, it comes with a number of unresolved issues such as security, scalability and performance, which limit its effectiveness in real business models.

In this paper we studied the relation between cybersecurity and blockchain. Specifically, we evaluated whether the technology represents an opportunity to improve recent cybersecurity challenges and limitations. We firstly defined a taxonomy of the four most prominent blockchain platforms and their consensus protocols, namely Bitcoin with PoW, Ethereum with PoS and PoA, and Hyperledger Fabric with PBFT. For our security analysis we divided the blockchain in a three-layer architecture, and we focussed on the two bottom layers: the network and consensus layers. In §3, we defined the most relevant security properties to matter in blockchain systems, and we used such properties to evaluate consensus protocols and platforms. Specifically, we evaluated the consensus resilience of each protocol by measuring their safety and liveness properties. For the evaluation of the blockchain platforms we instead considered the well known CIA triad, and we evaluated whether the platforms satisfy its properties, in addition to the so called '*profiling*' properties of accountability and authorisation. In this analysis we also introduced an important property, called *fairness*, for both the consensus algorithms and the platforms. It emerges from the results illustrated in §4 that permissioned blockchain like Hyperledger Fabric and Ethereum-PoA can guarantee good fairness and trustworthiness, and thus provide properties such as accountability, confidentiality and client fairness. However, these systems may require strong assumptions on the underlying network and the number of possibly subverted nodes in order to guarantee integrity and availability. Conversely, permissionless platforms as Bitcoin and Ethereum-PoS offer better integrity and availability, despite failing on profiling, confidentiality and fairness properties.

Following the results of our analysis, we studied the possible use of blockchain in five prominent cybersecurity topics such as data security, data privacy, trust, resilience and forensics. By referencing the same properties used in the security analysis, we identified the requirements underlying each of the topic, and the opportunities to integrate them with blockchain platforms studied in §4.

The impact of smart contracts. The current strong interest around blockchain has lead to the implementation of computing infrastructures based on decentralised applications. Those run on top of the application layer of blockchain and are implemented through the use of *smart contracts*. It is hard to understate the role of smart contracts, as they essentially make the blockchain data structure a programmable infrastructure for decentralised computation. Correspondingly, this application layer and the vulnerabilities that smart contracts introduce cannot be overlooked in the context of a broad security evaluation of blockchain platforms. Indeed, in the last few years, a number of attacks leveraging on weaknesses in the application layer have been perpetrated against the all the platforms we analysed in this paper. For example, in June 2016 the DAO governance application, developed on top of Ethereum public blockchain, has been hacked by abusing of a loophole in a smart contract. The attacker managed to retrieve approximately 3.6 million Ether [14]. In 2017, a critical vulnerability in the smart contract library for multi-signature wallets on Ethereum Parity was exploited by an attacker who was able to freeze 573 wallets [17]. Other critical vulnerabilities have been caused in Ethereum smart contracts through the introduction of hard forks, which caused some of the contracts stop working correctly [21, 13].

Against this backdrop, it appears clear that the thorough analysis of the security of smart contracts in blockchain platforms will require a dedicated study unfolding along three axes: blockchain platforms and cybersecurity properties, like in this paper, as well as language security properties. Such an analysis will almost certainly need to be based on a new approach extending the one we devised and adopted here, and is therefore outside the scope of this paper. We therefore leave the analysis of the security aspects of the blockchain application layer and a systematic evaluation of the smart contract attack surface to a forthcoming future paper.

References

[1] Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput. 1*, 1 (Jan. 2004), 11–33.

[2] Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., and Danezis, G. Consensus in the age of blockchains. *CoRR abs/1711.03936* (2017).

[3] Bistarelli, S., Mercanti, I., and Santini, F. A suite of tools for the forensic analysis of bitcoin transactions: Preliminary report. In *European Conference on Parallel Processing* (2018), Springer, pp. 329–341.

[4] BitFury Group. Incentive mechanisms for securing the bitcoin blockchain. *White Paper* (2015).

[5] BitFury Group, and Garzik, J. Public versus private blockchains part 1: Permissioned blockchains. *White Paper* (2015).

[6] BitFury Group, and Garzik, J. Public versus private blockchains part 2: Permissionless blockchains. *White Paper* (2015).

[7] Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., and Felten, E. W. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy* (2015), IEEE, pp. 104–121.

[8] Buterin, V. A proof of stake design philosophy. Medium Blog.

[9] Cachin, C., and Vukolic, M. Blockchain consensus protocols in the wild. *CoRR abs/1707.01873* (2017).

[10] Castro, M., and Liskov, B. Practical Byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation* (Berkeley, CA, USA, 1999), OSDI '99, USENIX Association, pp. 173–186.

[11] Chen, L., Xu, L., Shah, N., Gao, Z., Lu, Y., and Shi, W. On security analysis of proof-of-elapsed-time (poet). In *International Symposium on Stabilization, Safety, and Security of Distributed Systems* (2017), Springer, pp. 282–297.

[12] Clique consensus. https://github.com/ethereum/EIPs/issues/225.

[13] Coindesk. Ethereum?s istanbul upgrade will break 680 smart contracts on aragon. https://www.coindesk.com/ethereums-istanbul-upgrade-will-break-680-smart-contracts-on-aragon.

[14] Coindesk. Understanding the dao attack. https://www.coindesk.com/understanding-dao-hack-journalists.

[15] Costan, V., and Devadas, S. Intel SGX explained. *IACR Cryptology ePrint Archive 2016*, 086 (2016), 1–118.

[16] Cramer, R., Damgård, I. B., and Nielsen, J. B. *Secure multiparty computation.* Cambridge University Press, 2015.

[17] Cyber Security Southampton Blog. An exploited vulnerabiliy in the parity multi-sig library wallet. https://medium.com/cybersoton/are-blockchains-really-safe-assessing-the-security-of-consensus-schema-84a838458aa3.

[18] De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., and Sassone, V. PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain. In *Proceedings of the Second Italian Conference on Cyber Security, Milan, Italy, February 6th - to - 9th, 2018.* (2018), E. Ferrari, M. Baldi, and R. Baldoni, Eds., vol. 2058 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[19] Deng, M., Wuyts, K., Scandariato, R., Preneel, B., and Joosen, W. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering 16*, 1 (2011), 3–32.

[20] Dwork, C., Lynch, N., and Stockmeyer, L. Consensus in the presence of partial synchrony. *J. ACM 35*, 2 (Apr. 1988), 288–323.

[21] Ethereum Blog. Security alert: Ethereum constantinople postponement. https://blog.ethereum.org/2019/01/15/security-alert-ethereum-constantinople-postponement/.

[22] Ethereum community. Ethereum Proof-of-Stake. https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ.

[23] Ethereum Consortium. https://www.ethereum.org.

[24] Eyal, I., and Sirer, E. G. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security* (2014), Springer, pp. 436–454.

[25] Francez, N. *Fairness.* Springer-Verlag, Berlin, Heidelberg, 1986.

[26] Gaetani, E., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., and Sassone, V. Blockchain-based database to ensure data integrity in cloud computing environments. In *ITA-SEC* (2017), vol. 1816, CEUR-WS.org.

[27] Gentry, C., and Boneh, D. *A fully homomorphic encryption scheme*, vol. 20. Stanford University Stanford, 2009.

[28] Gheorghiu, V., Gorbunov, S., Mosca, M., and Munson, B. Quantum proofing the blockchain. *Blockchain Research Institute. November* (2017).

[29] Go Ethereum (Geth). `https://geth.ethereum.org`.

[30] Herlihy, M., and Moir, M. Enhancing accountability and trust in distributed ledgers. *CoRR abs/1606.07490* (2016).

[31] Hileman, G., and Rauchs, M. 2017 global blockchain benchmarking study, 2017.

[32] Hyperledger Fabric. `https://www.hyperledger.org/projects/fabric`.

[33] Karame, G. O., Androulaki, E., Roeschlin, M., Gervais, A., and Čapkun, S. Misbehavior in bitcoin: A study of double-spending and accountability. *ACM Trans. Inf. Syst. Secur. 18*, 1 (May 2015), 2:1–2:32.

[34] Lamport, L. Paxos made simple, fast, and Byzantine. In *Procedings of the 6th International Conference on Principles of Distributed Systems. OPODIS 2002, Reims, France, December 11-13, 2002* (2002), A. Bui and H. Fouchal, Eds., vol. 3 of *Studia Informatica Universalis*, Suger, Saint-Denis, rue Catulienne, France, pp. 7–9.

[35] Lamport, L., Shostak, R., and Pease, M. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS) 4*, 3 (1982), 382–401.

[36] Linden, T., Khandelwal, R., Harkous, H., and Fawaz, K. The privacy policy landscape after the gdpr. *arXiv preprint arXiv:1809.08396* (2018).

[37] McConaghy, T., Marques, R., Müller, A., De Jonghe, D., McConaghy, T., McMullen, G., Henderson, R., Bellemare, S., and Granzotto, A. Bigchaindb: a scalable blockchain database. *white paper, BigChainDB* (2016).

[38] Miller, T., Schelp, B., and Duncan, P. Systems and methods for managing database authentication and sessions, Nov. 9 2010. US Patent 7,831,616.

[39] Möser, M. Anonymity of bitcoin transactions an analysis of mixing services, 2013.

[40] Nagaraja, A., Mangathayaru, N., and Rajashekar, N. Privacy preserving and data security - a survey. In *2016 International Conference on Engineering MIS (ICEMIS)* (Piscataway, NJ, USA, 2016//), pp. 6 pp. –. privacy preserving;data security;crypto security features;public keys;private keys;data privacy;cryptographic algorithm;data transmission;.

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

[42] Nxt community. Nxt Whitepaper, 2014.

[43] Oki, B. M., and Liskov, B. H. Viewstamped replication: A new primary copy method to support highly-available distributed systems. In *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing* (New York, NY, USA, 1988), PODC '88, ACM, pp. 8–17.

[44] Olson, K., Bowman, M., Mitchell, J., Amundson, S., Middleton, D., and Montgomery, C. Sawtooth: An introduction. *The Linux Foundation, Jan* (2018).

[45] Parity Ethereum. A blockchain infrastructure for decentralised web, 2018.

[46] Parity Technologies. Aura - authority round consensus. https://wiki.parity.io/Aura.

[47] Peercoin. https://peercoin.net.

[48] Pomroy, S. P., Lake, R. R., and Dunn, T. A. Data masking system and method, July 5 2011. US Patent 7,974,942.

[49] Ponemon Institute. 2018 cost of a data breach study: Global overview. https://databreachcalculator.mybluemix.net/assets2018_Global_Cost_of_a_Data_Breach_Report.pdf, 2018.

[50] Prahlad, A., and Ngo, D. Systems and methods for performing data replication, Jan. 26 2010. US Patent 7,651,593.

[51] Robling Denning, D. E. *Cryptography and data security.* Addison-Wesley Longman Publishing Co., Inc., 1982.

[52] Shmueli, E., Vaisenberg, R., Elovici, Y., and Glezer, C. Database encryption: an overview of contemporary challenges and design considerations. *ACM SIGMOD Record 38*, 3 (2010), 29–34.

[53] Van Blarkom, G., Borking, J. J., and Olk, J. E. Handbook of privacy and privacy-enhancing technologies. *Privacy Incorporated Software Agent (PISA) Consortium, The Hague 198* (2003).

[54] Van Dijkhuizen, N., and Van Der Ham, J. A survey of network traffic anonymisation techniques and implementations. *ACM Computing Surveys 51*, 3 (2018/07/), 52 (27 pp.) –.

[55] Vukolić, M. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International Workshop on Open Problems in Network Security* (2015), Springer, pp. 112–125.

[56] Vukolic, M. Eventually returning to strong consistency. *IEEE Data Eng. Bull. 39* (2016), 39–44.

[57] Vukolić, M. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *Open Problems in Network Security* (Cham, 2016), J. Camenisch and D. Kesdoğan, Eds., Springer International Publishing, pp. 112–125.

[58] Weber, I., Gramoli, V., Ponomarev, A., Staples, M., Holz, R., Tran, A. B., and Rimba, P. On availability for blockchain-based systems. In *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)* (2017), IEEE, pp. 64–73.

[59] Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper* (2014).

[60] Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., Pautasso, C., and Rimba, P. A taxonomy of blockchain-based systems for architecture design. In *2017 IEEE International Conference on Software Architecture (ICSA)* (April 2017), pp. 243–252.