



Security and dependability analysis of blockchain systems in partially synchronous networks with Byzantine faults

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

To cite this article: Stefano De Angelis, Federico Lombardi, Gilberto Zanfino, Leonardo Aniello & Vladimiro Sassone (24 Oct 2023): Security and dependability analysis of blockchain systems in partially synchronous networks with Byzantine faults, International Journal of Parallel, Emergent and Distributed Systems, DOI: [10.1080/17445760.2023.2272777](https://doi.org/10.1080/17445760.2023.2272777)

To link to this article: <https://doi.org/10.1080/17445760.2023.2272777>



© 2023 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.



Published online: 24 Oct 2023.



Submit your article to this journal [↗](#)



Article views: 866



View related articles [↗](#)



View Crossmark data [↗](#)



Citing articles: 1 View citing articles [↗](#)

Security and dependability analysis of blockchain systems in partially synchronous networks with Byzantine faults

Stefano De Angelis^a, Federico Lombardi^{a,b}, Gilberto Zanfino^a, Leonardo Aniello^a and Vladimiro Sassone^a

^aSchool of Electronics and Computer Science, University of Southampton, UK; ^bConio Inc., San Francisco, CA, USA

ABSTRACT

Blockchains enhance trust in decentralized systems through components like distributed consensus, peer-to-peer communication, and trustless computing. Deployed in real networks, they must ensure security and dependability at each level. This paper introduces security and dependability concepts in three key layers: consensus protocols, network infrastructures, and smart contract applications, derived from distributed systems theory and adapted to blockchains. It evaluates these attributes across eight famous blockchains, assuming realistic network deployments and Byzantine fault scenarios to reveal system strengths and weaknesses. The aim is to establish a comprehensive framework for blockchain assessments, providing insight into design choices and potential security issues.

ARTICLE HISTORY

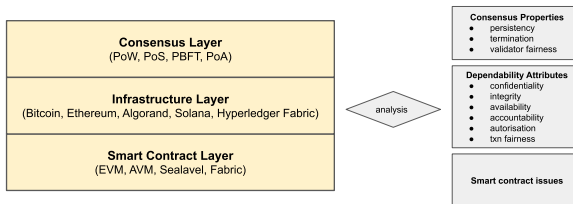
Received 6 February 2023
Accepted 3 October 2023

KEYWORDS

Blockchain; security; dependability; consensus; Byzantine fault tolerance; smart contracts security

Security and Dependability Analysis of Blockchain Systems in Partially Synchronous Networks with Byzantine Faults

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



1. Introduction

The Blockchain is a disruptive technology promising to evolve traditional centralised infrastructures, into decentralised collaborative systems. Such a change of paradigm is the core element of the emerging *web3*, where the web application concept is evolving towards decentralised application (DApp). In this new scheme, rather than running the business logic of software on a cluster of servers owned by a single entity, the blockchain is used as a democratic platform that can be publicly joined by users to act either as a client – which simply uses software running on top of a blockchain – or a server, i.e. a node of the blockchain which maintains the ledger and confirms transactions and blocks. The advantages and

CONTACT Stefano De Angelis  s.deangelis@soton.ac.uk  University of Southampton, University Rd, Southampton SO171BJ, UK

© 2023 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License (<http://creativecommons.org/licenses/by-nc-nd/4.0/>), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited, and is not altered, transformed, or built upon in any way. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

the success of decentralisation are mainly due to removing the single point of failure, reducing trust requirements, and providing a robust infrastructure harder to compromise [1]. However, blockchains have suffered from security and usability flaws that have limited their wider adoption. For instance, in 2021 about US\$ 1.3B got stolen from decentralised finance (DeFi) protocols [2] by exploiting code issues related to smart contracts, i.e. programmes deployed and executed on top of a blockchain. It is important to highlight that security flaws may go beyond smart contracts. For instance, the blockchain Solana experienced several outages in the past years caused by consensus issues – Solana’s validators are publicly known and deterministically inferred by protocol. For example, in 2021s incident [3], a Distributed Denial-of-Service (DDoS) attack was conducted against the validators, leading to a network outage. Also, in 2010 the Bitcoin network at block 74638 experienced a protocol violation which allowed two addresses to receive 185 million Bitcoins [4], i.e. about nine times the maximum supply defined by the protocol (21 million). Such an issue was caused by a value overflow vulnerability in the Bitcoin core software, not the protocol. Those examples explain why blockchain protocols security-by-design are difficult to guarantee at scale due to issues that can be found at different architectural layers. For this reason, it is fundamental to shed light on the security aspects of any blockchain components.

In a distributed system, security is often incorporated within *dependability* [5,6]. A dependable system must ensure a set of attributes including, among others, reliability, availability, confidentiality, and integrity. Many works attempted to evaluate these properties for blockchains. However, most of the studies focus their analysis on the blockchain consensus protocols [7–9], and lack of a comprehensive evaluation. Beyond the consensus, blockchains entail other components at the infrastructure and application layer. Assessing the security and dependability of the overall blockchain system is not trivial since each component must be properly considered. Furthermore, evaluating a distributed system implies considering realistic assumptions, therefore for a blockchain intended to work on the Internet, it becomes mandatory to consider a partially synchronous network with possible Byzantine faults. This paper defines the concepts of security and dependability for blockchain. It traces a line about which attributes must be considered to assess among others, the trustworthiness and reliability of a specific blockchain protocol. At first, the considered security and dependability attributes are presented. They have been derived from existing attributes previously defined in the theory of distributed systems [5,6]. The novel attributes take into account crucial concepts for blockchains, such as forks, i.e. eventual splits of the ledger, delayed block confirmation times, and transactions finality [7,9]. The identified properties are thus evaluated for three blockchain architectural layers: the *consensus protocol*, the *network infrastructure*, and the *smart contract applications*. The analysis considers eight blockchain platforms, namely *Bitcoin* [10], *Monero* [11], *Ethereum* (with its new consensus protocol update) [12], *Algorand* [13], *Solana* [14], *Hyperledger Fabric* [15] and two private instances of *Ethereum*, referred as *Ethereum-private* [16]. Platforms and protocols are presented first, as well as a list of 15 known *Ethereum* smart contract vulnerabilities. To the best of our knowledge, this is the first analysis that demonstrates how different consensus designs, network models, and smart contracts issues may independently affect the whole system. The contributions of this work are summarised as follows:

- *Security and dependability attributes for blockchains*: set of novel attributes refined from foundational concepts adopted in the field of distributed systems; introduction of three novel attributes relevant for blockchains, such as accountability, authorisation, and fairness;
- *Three layers analysis*: first security and dependability evaluation across three blockchain architectural layers, such as the consensus protocol, network infrastructure, and smart contract applications. The analysis assumes a realistic, internet-based, network deployment with a partially synchronous model and Byzantine faults.
- *Smart contract vulnerabilities analysis*: analysis of 15 well-known *Ethereum* smart contract vulnerabilities with respect to other emergent blockchain platforms, such as *Algorand*, *Solana*, and *Hyperledger Fabric*.

This work's aim is to lay the basis for further security and dependability studies of blockchains, providing the foundational attributes to consider both with qualitative and quantitative evaluations.

Paper structure. The paper is organised as follows. Section 2 discusses related work, Section 3 introduces the blockchain platforms we consider in our study, Section 4 describes their underlying consensus protocols, whereas Section 5 presents a collection of Ethereum smart contract vulnerabilities. Then, Section 6 defines the refined security and dependability properties and proposes the security analysis of them at consensus, network infrastructure, and smart contract applications layers. Finally, Section 7 sums up the results drawing the line for future directions.

2. Related work

In literature, some effort has been devoted to studying the security and dependability of consensus protocols employed for blockchain systems [7–9]. However, most of these studies focus only on a particular blockchain component or platform. For instance, Cachin et al. in [7] provide an assessment of safety and liveness properties for blockchain consensus protocols in permissioned blockchains. Differently, in [17–19] the authors provide a survey of known vulnerabilities for Ethereum smart contract, however, none of them evaluate the vulnerabilities on other platforms. Therefore, a fair comparison of blockchain security and dependability becomes elusive due to several contrasting assumptions. To evaluate blockchain systems, Kannengießer et al., provide in [20] a classification of the most relevant properties and their tradeoffs. The authors state that security and performance exhibit more tradeoffs with respect to security and performance, and thus are classified as the more relevant properties in a blockchain system. In Xiao et al. [21], present a comprehensive analysis of the performance and security of the most prominent blockchain consensus protocols. The authors introduce an evaluation framework based on five components of a blockchain system that impact the consensus. Hence, they use such a framework to compare the protocols with respect to different performance metrics. A qualitative approach is also used in [22] by *Mingxiao et al.*, to extensively compare blockchain protocols like PoW, PoS, DPoS, PBFT, and Raft, and their performance and security properties. Similarly, *Sankar et al.*, investigate in [23] the main differences between SCP – Stellar Consensus Protocol – and the consensus algorithms employed in R3 Corda and Hyperledger Fabric. *Kiffer et al.*, propose in [24] a formal approach to analyse the consistency properties of blockchains by applying a Markov-chain based method. The study analyses a series of attacks on the protocols and evaluates how those attacks affect the protocols. In Pass et al. [25], formally prove that PoW-based consensus protocols can guarantee consistency and liveness even in asynchronous networks, while in [26,27], the author demonstrates a security flaw in the first open-source implementation of the IBFT protocol. These works present a formal definition of the IBFT and demonstrate that in the presence of a partially synchronous network, the protocol is not BFT, violating both safety and liveness. Similarly, in [28], the authors analyse the correctness of Tendermint – a consensus protocol that combines the PBFT voting mechanism with the PoS validators election. The authors claim that Tendermint satisfies the consensus security properties under the assumptions of $f < n/3$ Byzantine nodes, but they state also that the leader election algorithm is not fair. Differently, *Ekparinya et al.* [29], evaluate the security of two PoAs consensus algorithms by means of experimental evaluation. The authors reproduce a double-spending attack and claim that PoAs are vulnerable to such an attack. PoA's performance and security tradeoffs have been also evaluated in [30] where the authors propose a CAP theorem-based comparison of AuRa and Clique, two leader rotation protocols, against the well-known PBFT.

In this paper, we aim to extend the state-of-the-art by providing an approach that goes beyond typical analysis; to the best of our knowledge, this is the first work evaluating the security and dependability of blockchains across three principal components of consensus, network infrastructure, and smart contract applications.

3. Blockchain platforms

This section introduces eight famous blockchain platforms, distinguishing them from public-permissionless and private-permissioned settings. The descriptions present the platforms' main technological advances and position them with respect to the *scalability trilemma* [31], i.e. the security, scalability, and decentralisation tradeoff.

3.1. Bitcoin

Bitcoin [10] is the first, open-source, decentralised, and permissionless blockchain enabling trustless electronic *machine-to-machine* payments without intermediaries. The Bitcoin network uses a distributed consensus protocol, the *Proof of Work (PoW)*, to ensure a unique ledger of payment transactions across the nodes, and to avoid issues like double-spending [10,32]. The hashing algorithm used by Bitcoin is SHA-256d, i.e. the standard SHA256 where the 'd' denotes a double of hash iterations. Therefore, the ASICs result in being the most efficient hardware for Bitcoin mining. Although PoW ensures nodes agree on the same order of transactions, it uses a probabilistic approach to elect block proposers with mining [9] which might cause conflicts. Indeed, it might happen that two miners are simultaneously elected as block proposers. In that situation, the blockchain forks and therefore consistency across all replicas cannot be guaranteed. To mitigate forks, Bitcoin adopts the longest chain rule in which one block is finalised, hence cannot be reverted [9], once six further blocks are proposed on top. The rationale behind this rule is that it is practically unfeasible for an attacker to revert six blocks. To ensure the correctness of accounts' balances, Bitcoin uses the UTXO model in which each transaction is composed of a list of unspent transaction outputs that indicate the balance of blockchain accounts. Despite Bitcoin's aim for decentralisation, its particular design induces miners to centralisation, the *mining pools* phenomenon [33], to increase their mining power. Finally, to ensure strong (eventual) consistency of the ledger, Bitcoin sacrifices performance: the throughput is only about 5 transactions per second (TPS) with a block confirmation of about 10 min [9].

3.2. Monero

Monero [11] is one of the principal and older blockchains focussed on privacy. It inherited the main logic of Bitcoin operation, indeed it is PoW-based with a UTXO transaction model, but it introduced the CryptoNote protocol [34] to limit the traceability of transactions. Specifically, Monero makes use of Ring Signatures to protect the sender's privacy and Ring Confidential Transactions (RingCT) to hide transferred amounts. The two solutions together make transactions indistinguishable from one another. Finally, to enhance the receiver's privacy, Monero uses Stealth Addresses, i.e. one-time public keys (OTPKs, also known as random one-time addresses) created by the sender on behalf of the recipient when a transaction is initiated. The output received by the recipient is not publicly associated with the related wallet address, but the sender can prove to the recipient the transaction on-chain. Monero, similarly to Bitcoin, produces blocks within minutes (about 2 min) and its mining algorithm is RandomX (replacing the former one, CryptoNightR) has been designed to resist ASIC, therefore the token (XMR) can be mined with a consumer-grade CPU or GPU hardware.

3.3. Ethereum

Ethereum. It is the first fully programmable blockchain platform introducing *smart contracts*, i.e. immutable pieces of code deployed and executed autonomously on the so-called *Ethereum Virtual Machine (EVM)* [35]. Ethereum smart contracts can be written in Turing-complete programming language like SOLIDITY [36] or VYPER [37]. The first version of Ethereum was based on the PoW consensus, like Bitcoin, but through the Ethash algorithm, i.e. an updated version of the Dagger-Hashimoto algorithm [38,39] which requires high memory and makes it therefore ASIC-resistant. Ethereum PoW

requires a shorter confirmation time compared to Bitcoin (about 14 s) which increases the throughput to about 30 TPS [40]. Despite the better performance, Ethereum is more prone to forks than Bitcoin and therefore consistency breaks are more frequent. In that case, the platforms adopt an algorithm called GHOST to recover from forks and establish a unique ledger across replicas [41]. Ethereum adopts a *account-based* model in which for each account the EVM provides a key/value store keeping track of balances. The EVM applies operational fees to process transactions measured as Ethereum *gas*. Transactions' *gas price* define the amount of ETH (the Ethereum's cryptocurrency) to be paid for each computational step, while *gas limit* is a scalar value representing the total amount of gas that can be consumed in a block by transactions.

Ethereum 2.0. It is the most important update of the Ethereum protocol to cope with scalability and performance issues. Among others, it proposes two major improvements, such as the shift from PoW to the *Proof of Stake (PoS)* implementation called *Casper FFG* [42], and the implementation of *Shard Chains*. The upgrade to PoS evolves Ethereum to a more energy-efficient platform, while Shard Chains may drastically improve scalability by changing the way the blockchain is replicated across the nodes of the network. Sharding allows parallel execution, enabling the achievement of better throughputs. However, it comes at a security cost since each shard is not managed by the entire network and hence is more vulnerable.

Ethereum Private Networks. Many clients of the Ethereum protocol can be used to set up a private network. We refer them to as *Ethereum-private* networks. Two of the most common Ethereum clients are *Geth* [43], the Ethereum implementation in GOLANG language, and *Parity* [44], a RUST-based implementation. *Ethereum-private* networks enable the integration of pluggable lightweight consensus algorithms different from PoW and PoS. These types of chains sacrifice decentralisation in favour of higher performance (thousands of TPS) and privacy guarantees [45,46]. The security of Ethereum-private does not depend on computational power, but rather on the number of nodes, an attacker can control. Usually, in such private networks, an adversary needs to control at least 1/3 of nodes to take control, although a wrong consensus implementation may drastically increase the probability of attack success [30].

3.4. Algorand

Algorand [47] is a novel permissionless blockchain platform that aims at solving the scalability trilemma. Algorand embeds a distributed computation engine, i.e. the *Algorand Virtual Machine (AVM)* which runs on every node of the network. The AVM interprets and executes self-verifiable, error-free smart contracts that automatically approve or reject transactions according to built-in business logic. Algorand smart contracts are written in an assembler-like, Turing-complete language called TRANSACTION EXECUTION APPROVAL LANGUAGE (TEAL). Similarly to Ethereum, Algorand's data model is account-based, however for each smart contract, there exist different types of storage. In particular, Algorand smart contracts can access limited global storage and per-account local storage for *key-value* store data, and then unlimited Box storage for random-sized byte arrays. Algorand offers unique features at the consensus level such as a tokenisation framework, *Algorand Standard Assets* and native atomic swaps, Atomic Transfers, which enable fast end secure development without introducing complexity. Algorand's core innovation is its consensus protocol, the Pure Proof of Stake (PPoS) [13], which ensures consensus agreement in a large-scale decentralised network without giving up neither scalability nor security. Algorand blockchain is designed not to fork ever, transactions are considered final as soon as executed and included in a block. This makes Algorand much faster than Ethereum with a block time of about 3.7 s and throughput of about 6000 TPS [47]. Algorand's protocol is considered safe until 3/4 of the total stake is kept honest [13].

3.5. Solana

Solana [14] is a permissionless blockchain platform whose main purpose is the performance enhancement of large-scale decentralised networks. It promises to achieve more than 50,000 TPS through

nodes' synchronisation. Transactions order is indeed guaranteed by a global and verifiable clock defined with the *Proof of History (PoH)*. Thanks to PoH, Solana relies on network synchrony assumptions to implement many performance enhancements, i.e. the adoption of fast transaction confirmations with optimistic finality, the speed up block propagation with a more efficient distribution protocol, and the removal of in-memory queued transactions. Solana also offers a parallel processing runtime for smart contracts, called *SEALAVEL*. Differently from the EVM, *SEALAVEL* can process tens of thousands of concurrent programme executions using as many cores as available by validators. Despite the outstanding performance, Solana sacrifices both security and decentralisation. The Proof of Stake consensus protocol elects block proposers from a well-known, deterministically computed, set of validators, which represent a point of vulnerability both for censorship resistance and availability guarantees.

3.6. Hyperledger fabric

Hyperledger Fabric [15] is a permissioned blockchain platform featured by a modular architecture. The distinguishing characteristic of Fabric is that it splits the transaction order, i.e. the consensus process, from transaction execution, i.e. the operations on users' assets. The assets within the ledger state are represented as a collection of *key-value* pairs, and through smart contracts (called *chain-codes* in Fabric's jargon), it is possible to combine their values to carry out complex functions according to users' needs, e.g. to perform an auction. Being permissioned, Fabric offers an *authentication* layer that identifies the system entities by issuing X.509 digital certificates. Additionally, the authentication process enforces authorisation policies on the operations. Fabric introduces the concept of *channels* to represent restricted consortium networks. Transactions within a channel remain private and shared only across channel participants, enabling data isolation and confidentiality. As the operating environment is more trusted than a permissionless setting, it allows employing a lighter consensus, which results in better performances despite restricted security assumptions.

4. Blockchain consensus protocols

This section introduces the consensus protocols of the blockchain platforms presented in Section 3. We present how these protocols work, by focussing the analysis on three principal characteristics (i) the trust assumptions under which each protocol operates, (ii) how the block proposers are elected as leaders, and (iii) how protocols behave under adversarial conditions.

4.1. Proof-of-work

The PoW [10] is a probabilistic consensus protocol consisting of computationally-intensive hashing tasks executed by some distinctive network nodes, called miners. Specifically, miners compete to find the solution to a complex cryptographic puzzle. Essentially, miners keep looking for a value of which hash is lower than a certain target number. Such a problem is hard to solve and requires miners to iterate over several computation steps. The miner who solves the puzzle is elected as leader and is allowed to propose a new block. PoW introduces randomness in leader election making it impossible to predict which miner will be elected as leader in the next protocol steps. For this reason, the PoW leader election is said probabilistic. The likelihood of solving a puzzle, hence speeding up the block production, is determined by the target number. Such a number is adjusted over time according to the desired *difficulty*. The more computational power available in the network, the higher difficulty will be configured by the PoW. Due to the probabilistic nature of the PoW, there can be simultaneous leaders proposing blocks concurrently. In this scenario, the blockchain will fork. As already mentioned before, the standard approach used by Bitcoin and Ethereum to solve forks is the longest-chain rule. The verification and subsequent acceptance procedures happening in PoW make blocks *persistent* unless an attacker controls most of the miners' hash power (the aforementioned 51% attack) [48,49]. Indeed, under the assumption of an eventually synchronous network, PoW ensures that, until there is

hash power available, it successfully proceeds. Also, being based on computational power rather than nodes, PoW is not vulnerable to *sybil attacks*. Although PoW offers strong integrity properties, besides energy inefficiency, it suffers from *performance* limitations due to the intensive hashing tasks required to generate randomness in leader election.

4.2. Proof of stake

The PoS [50] is the main alternative to PoW for permissionless blockchains. It replaces the computational requirements used in PoW with users' money commitment, i.e. *stake* deposit. The PoS security does not depend on protocol rewards, like mining rewards in PoW, but on penalties. Users commit their money '*at-loss*' in order to become blockchain validators. PoS validators are special users in charge of producing and validating blocks; validators' power is proportional to the stake they deposit. The more stake is committed, the greater the incentives to behave honestly. Malicious validators are penalised by protocol having some of their capital removed forever. This phenomenon is known as *slashing*.

Although the PoS works under the same principle of putting capital 'at risk', there exist several variants which differ from the performance and security degrees. Indeed having large-scale validator sets results in more decentralised and secure systems but introduces several scalability limitations [51]. The next section presents three different PoS variants, respectively adopted by the Ethereum, Algorand, and Solana permissionless blockchains.

4.2.1. Ethereum proof of stake

The *Ethereum PoS (EPoS)* protocol has been implemented with the algorithm *Casper FFG* [42]. It ensures that, until a super-majority of the stake is honest, the network agrees on the same finalised view of the ledger in a finite time-bound. The algorithm is organised in three phases, namely *proposal*, *justification*, and finalisation. Each phase lasts for a prefixed period of time called *epoch*, and each epoch is divided into smaller time intervals called slots. Over the epochs, the validators agree on the blocks proposed by a validator leader randomly elected. Blocks get finalised once the majority of validators, i.e. 2/3 of the total committed stake, agree on the same state for two consecutive epochs, i.e. justification and finalisation phases. Any user can join the network as a validator by committing 32 ETH¹ at stake, and Byzantine behaviour is punished by the protocol by slashing the validators' entire stake. Conversely, to PoW, the Ethereum PoS protocol causes no waste of energy since it does not require computational tasks to be solved, therefore, performance can be much better. Attacking a PoS requires an attacker to control the majority of committed staking, making it not vulnerable to *sybil attacks*.

4.2.2. Algorand proof-of-stake

The *Algorand PoS (APoS)* protocol is called Pure Proof of Stake (PPoS) [13]. It leverages the cryptographic primitive known as VRF (Verifiable Random Functions) [52] to significantly decrease the high volume of exchanged messages occurring in traditional BFT protocols [53]. Specifically, PPoS works as follows: it proceeds in rounds, and for each round, there are three phases: *block proposal*, *soft vote*, and *certify vote*. When a round starts, users use the VRF to select themselves as leader and committee members. In the *block proposal* phase, the leader selected by the VRF propagates a new block along with the VRF output, which proves that the account is a valid proposer. Then in the *soft vote*, a selected committee of users cast a vote on the block proposals. This phase reduces the number of proposals down to one, guaranteeing that only one block gets certified in a round. When a quorum of votes from the committee members is reached starts the *certify vote*. In this last phase, a new committee checks the validity of the block at the *soft vote* stage. Thus a new vote begins to certify the block. When a quorum is reached, the block is committed and the round terminates. Each phase is characterised by a timeout to ensure safety when partitions occur. PPoS can achieve higher throughput and lower block time than traditional PoS due to a reduced message exchange. Furthermore, the VRF makes the leader unpredictable, dismissing the possibility of having fixed validators such as in traditional PoS.

4.2.3. Solana proof-of-stake

The foundational primitive of the *Solana PoS (SPoS)* protocol is the PoH, a cryptographic function enabling the definition of a common source of synchronisation across the network. PoH has been implemented on the idea of VDF (Verifiable Delay Function) [54], i.e. a cryptographic function providing verifiable proof that some time has passed. However, PoH's verification is algorithmically slow and should not be confused with classical VDFs [55]. The PoH provides a global clock used by the Solana network to coordinate the PoS consensus protocol and reduce message overheads that affect classical BFT protocols [53]. The SPoS protocol proceeds in steps of fixed length dictated by the PoH. For each step of the PoH, a leader is elected as a block proposer. Leaders are elected by running the Leader Schedule Rotation, a deterministic protocol that, given a list of validators, precomputes in advance the list of leaders that will be elected according to their stake [56,57]. This protocol enables each validator to be aware of the next leader proposing a new block in a certain PoH time. Under the assumption of a supermajority of honest stake ($> 2/3$ total stake), Solana provides optimistic finality, i.e. validators start building their own fork of the blockchain independently, and if the supermajority of the stake adheres to the same fork, it can be considered final. To vote on a certain fork Solana applies an efficient BFT protocol called *TowerBFT*, which ensures the achievement of finality in seconds. PoH is also used to checkpoint the state of the whole Solana blockchain. Data is indeed timestamped into the PoH (hash of ledger data) to guarantee that the data itself was created in a precise order. As other PoSs, SPoS ensures requires an attacker to control the majority of committed stake, making it not vulnerable to *sybil attacks*.

4.3. Practical Byzantine fault tolerance

The *PBFT* consensus protocol [53] is characterised by a single-leader and view-change approach. 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 the total order of transactions. In case of misbehaving leaders, 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, the PBFT consensus protocol guarantees strong consistency provided that $f < N/3$, with f malicious nodes and N replicas. PBFT has been proven to be optimal with $N \geq 3f + 1$ nodes [53]. In a permissionless setting, i.e. where nodes are not authenticated and can behave arbitrarily, PBFT is vulnerable to sybil attacks though since it cannot distinguish if an attacker is falsely impersonating multiple nodes. For this reason, this consensus algorithm can be safely used only in permissioned settings.

4.4. Proof-of-authority

The *Proof-of-Authority (PoA)* has been proposed as part of the Ethereum consortium for private networks and implemented with two protocols called *AuRa* and *Clique* [58,59]. PoA relies on a set of trusted authorities running the consensus. Consensus in PoA relies on a *leader rotation* schema, which distributes the responsibility of block creation among the authorities [60,61]. Time is divided into *steps*. In each step, an authority is elected as the proposer. The way authorities are elected differs in the two consensus protocols. AuRa proposes a deterministic function based on UNIX times, which requires strong synchronisation assumptions. Conversely, Clique computes leaders according to the number of the next block on the blockchain. The PoA is a hybrid consensus protocol between the lottery-based and voting-based approaches. PoA protocols guarantee eventual consensus on transactions. Indeed, the lightweight leader election may lead to forks that eventually get resolved. Consequently, PoA cannot achieve instant finality but this is delayed in time. According to the concept of the longest chain, a block in PoA is considered final when a majority of further validators have proposed a new block in their turn, under the assumption that blocks are proposed at a constant rate [58]. These algorithms are vulnerable to Sybil attacks, and thus cannot be used in permissionless settings.

5. Smart contracts vulnerabilities

Smart contracts are computer programmes executed on the blockchain. These programmes compile to bytecode instructions that are then processed on a decentralised execution environment (e.g. Ethereum's EVM). Therefore, applications built on the blockchain must entrust the smart contracts layer to provide secure and dependable services. A security flaw or issue at this layer can potentially break the entire application and eventually lead to economic or information loss.

This section provides a list of well-known smart contract vulnerabilities that have been exploited on Ethereum. We refer as smart contract *vulnerability* to any issue generated at the source code level (smart contract code), execution engine level (bytecode), or blockchain level. A similar classification has been also given in [18]. The list has been selected from a review of the state-of-the-art of Ethereum smart contracts vulnerabilities [17–19,62,63], and open source projects like the Decentralised Application Security Project (DASP) [64], and the EEA EthTrust Security Levels Specification [65]. The mapping has been conducted considering a list of 54 Ethereum vulnerabilities classified with the DASP Top 10 in [63]. Therefore, these have been filtered out following three criteria: relevance, accuracy, and applicability. Relevance and accuracy have been qualified by matching the vulnerabilities effectively exploited on existing smart contracts [17,19,62]. This allowed us to reduce the list to 22 vulnerabilities. Therefore, we analysed them with respect to their applicability to other blockchain execution environments. In particular, we identified 15 vulnerabilities that are exploitable or can be evaluated, on other blockchains. We exclude from the list those issues related to EVM-specific characteristics. Thus, the identified smart contract vulnerabilities are:

(V_1) *Reentrancy*. This vulnerability occurs when a caller contract invokes a function of an external callee contract. For example, a malicious actor can call back from the callee contract a funds withdrawal function of the caller contract before the execution of the caller contract terminates – this triggers an infinite loop of calls. Leveraging reentrancy, the attacker could bypass the validity checks of the caller contract and iterate infinitely. In Ethereum, the exploit of a reentrancy issue may lead to indefinite withdrawal calls. Two reasons cause this vulnerability [66]: (i) validity checks are handled by state variables that are not updated until other transactions terminate, (ii) no gas limit is required when handling interactions between external smart contracts. Prevention methods consist of (i) updating the state variables before calling external contracts; (ii) introducing a `mutex lock` [36] in the contract state so that only the owner of the lock can access and update the state.

(V_2) *Integer overflow and underflow*. This vulnerability occurs when a function computes an arithmetic operation that falls outside a specific datatype. A prominent role is played by the programming language, hence some protections exist both natively or through an external library. Ethereum does not provide native prevention for smart contracts, but some recommendation has been defined, such as (i) using *SafeMath* library [67] to check on underflow/overflow, (ii) using *Mythril* library [68] to check the security of EVM bytecode before its execution.

(V_3) *Frozen Token*. This vulnerability causes users' funds deposited in the contract account to be locked and impossible to withdraw back, effectively freezing them into the contract. Both Ethereum and Algorand are vulnerable to such an issue. The causes of this vulnerability are twofold: (i) the deposit contract account does not provide any function to spend funds using a function from an external contract as a library, (ii) the callee contract function (`selfdestruct` for Ethereum, `Delete` for Algorand and `solana programme close` command for Solana) is executed without checks. Prevention method [19,62]: a deposit contract shall assure that the mission-critical functions or money-spending functions are not outsourced to external contracts.

(V_4) *DoS with unexpected revert*. This issue occurs when the execution of a transaction is reverted due to a thrown error or a malicious callee contract that deliberately interrupts the execution. Ethereum's mitigation involves the use of execution rewards to discourage transactions from reverting.

(V₅) *DoS with GasLimit*. This vulnerability causes transactions to be aborted due to exceeding the gas limit. This affects Ethereum having functions consuming more computational resources (measured in Ethereum gas) than expected. To mitigate this issue the contracts should not execute loops on accounts accessible data structures. Loops should be controlled, such that the execution always terminates, even when transactions are aborted.

(V₆) *Insufficient signature information*. This vulnerability causes a digital signature to be valid for multiple transactions. This happens when a sender uses a *proxy* contract [19,62] as a deposit for one or more authorised receivers. An authorised receiver gets an off-chain, digitally signed message from a certain sender. Therefore the receiver withdraws funds from that proxy, which must verify the validity of the digital signature. If the signature is malformed (missing nonces, or proxy contract address), a malicious receiver could reuse the signature to reply to the withdrawal transaction and drain the proxy balance. To prevent this issue, the address and the nonce must be checked within signed messages.

(V₇) *Generating randomness*. This vulnerability concerns smart contracts using pseudorandom number generators (PRNG) to create random numbers for application-specific use cases. Specifically, this vulnerability affects methods using random numbers created by a PRNG, in which the base seed of the generator is a parameter controlled by miners, e.g. Solidity's `block.number`, `block.difficulty`. A malicious miner can manipulate the PRNG to generate an output that is advantageous for itself. A generally safe mitigation is to use the `blockhash` parameter of some blocks in the feature as a seed of a pseudo-random number generator [69]. Although also off-chain PRNG approaches can be used, e.g. the use of oracles [70].

(V₈) *Block Timestamp manipulation*. This vulnerability affects smart contracts that use `timestamp` parameter in the control-flow (e.g. for periodic payments) or as source of randomness [62]. As block proposers can control this parameter, they could adjust that value to change the logic of functions, and thus take profit. Ethereum is vulnerable to such issues and a prevention method consists of avoiding the parameter in any control-flow decision logic.

(V₉) *Front Running*. This vulnerability is caused by a malicious manipulation of the transaction priority mechanism. In Ethereum, validator tips can be added to a transaction to prioritise its execution over others [35]. Malicious block proposers might decide to alter the priority process by processing their preferred transactions. This exposes the system to unexpected manipulations of the blockchain state [19]. Front running is usually adopted by validators for MEV (Maximum Extracted Value) activities, such as maximising the rewards obtained from block production, and taking advantage of arbitrage opportunities from decentralised exchanges [71]. A mitigation method consists of obfuscating the priority fees to external observers using cryptographic commitments [62].

(V₁₀) *Under-priced opcodes*. This vulnerability is caused by under-priced opcodes that can be executed at a low cost while consuming a lot of resources. Malicious actors might trigger several invocations of these opcodes to inject high computational requirements and overload the network. This vulnerability affected Ethereum and was caused by misconfigured parameters [62]. The Ethereum protocol has been upgraded to limit opcodes under-pricing.

(V₁₁) *Token lost to orphan address*. This vulnerability is caused by a lack of validation checks on payment transactions. Ethereum only checks the recipient's address format but not whether such an address is valid or if it exists. If a user sends money to non-existing addresses Ethereum automatically creates a new *orphan* address [18]. The user does not have control of the orphan address, hence it is impossible to recover funds from there.

(V₁₂) *Short address*. This vulnerability affects only Ethereum and it is caused by the EVM missing validation check on addresses. Recall that inputs are expressed as an ordered set of bytes, in which the first four bytes identify the callee's function, and then other inputs are concatenated in chunks of 32 bytes. In case of arguments with fewer bytes, EVM auto-pads with zeros at the end of the byte string filling the 32-byte chunk. An attacker could manipulate this process to execute malicious

actions. For instance, if we consider the `transfer(address addr, uint tokens)` function and a badly formatted `addr` with one missing byte, the auto-pad will add extra zeros at the end of `addr`, and consequently increase the number of tokens to transfer [62]. To prevent that, developers must prepare functions validating transaction input lengths.

(V_{13}) *Erroneous visibility*. This vulnerability affects only public blockchains. Transactions (including data, balances, and contract codes) are visible to external observers. Solidity provides four types of visibility to restrict access to Ethereum's contract functions, namely `public` open to everyone, `external` only externally from the contract, `internal` only internally (the contract and its related contracts), and `private` only within the contract. By default, Solidity assigns the type `public` to functions, hence in case of erroneous visibility configuration, an attacker might be able to call the function from the external [62]. To avoid this, with Solidity 0.5.0 and above, it is mandatory to specify the function visibility.

(V_{14}) *Unprotected suicide*. In Ethereum, Solidity contracts can be killed or deleted using the `suicide` or `self-destruct` methods. Usually, only the contract's owner, or authorised external users, can invoke these functions. However, there might be cases in which the owner is not verified by the functions, or the owner itself is malicious, in that case, an attacker can invoke one of these methods and kill the contract. As a prevention method, developers must enhance security checks with permissions mechanisms, assuring that contract destruction calls properly handle the correct closure (e.g. freeing up funds) and get approved by different authorised parties.

(V_{15}) *Unrequested Token*. Ethereum's ERC-20 tokens can be arbitrarily sent to any Ethereum address. Users do not need to provide any approval to receive tokens in their wallets. This is typically used as a scam or phishing attack. For example, malicious actors can mint ERC-20 tokens with malicious instructions injected into the contract to drain the users' wallets once triggered unintentionally [72].

6. Security and dependability evaluation

6.1. Overview

Dependability is the capability of a system to deliver trusted services in the presence of internal or external failures. Dependability embraces the availability, reliability, safety, and integrity attributes previously defined in [5,6]. The former two determine the capability of delivering a continuous service, while the latter two ensure correctness. Dependability is usually paired with *security* as a concept that combines confidentiality, i.e. the ability to not disclose information to unauthorised parties, to integrity and availability. Security and dependability also cope with the system's maintainability [5]. However, blockchain's maintainability depends on the utilised governance rules [73]. In this work, the governance and maintainability are considered out of scope. Therefore, we introduce three novel concepts, such as *accountability*, *authorisation*, and *fairness*. The first two are security attributes, and determine the permissions enforced by the blockchains to limit users' access and control [60,74–76]. Fairness, instead, delineates whether the system ensures equitable treatment to any actor or process, without giving an unfair advantage or disadvantage to a particular component [77]. We distinguish two fairness attributes for blockchains, namely *validator fairness* and *transaction fairness*. These attributes are then validated over the three architectural layers of *consensus protocols*, *network infrastructure*, and *smart contract application* layers of each blockchain and protocol presented in Sections 3 and 4. Figure 1 provides the classification overview.

The analysis provided in the following sections assumes blockchains to be deployed on a partially synchronous network [78] in which messages between nodes can be arbitrarily delayed but eventually delivered within a fixed time-bound. The analysis also considers the presence of Byzantine faults caused by faulty nodes or malicious actors. We refer to n as the number of nodes in the network that maintain a local copy of the blockchain, f as the Byzantine faulty nodes, and v as the subset of nodes, called validators, entitled to propose new blocks. Nodes append blocks on their local copy of the ledger

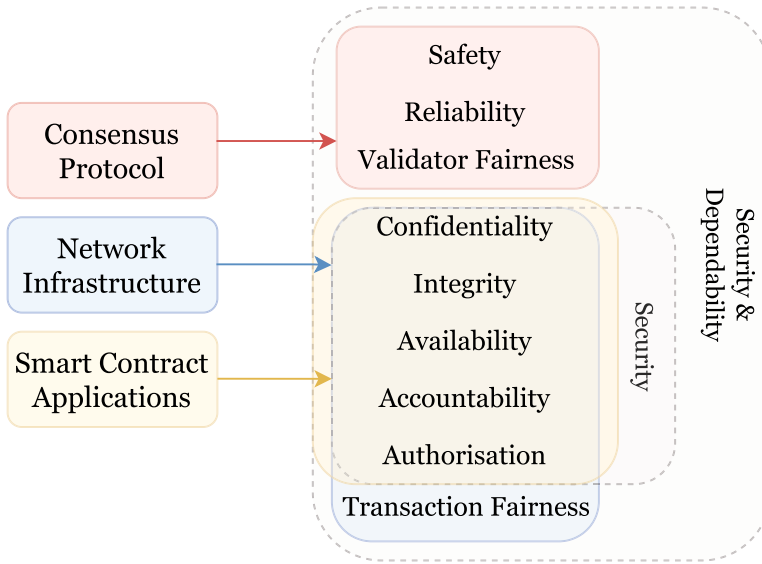


Figure 1. Security and dependability attributes classification.

once validators achieve consensus. We say that a block is appended at height h , when it is the h th block of the ledger.

6.2. Consensus protocols

Secure and dependable consensus protocols must guarantee *safety* and *liveness* [7] properties when operating under adversarial conditions. In the blockchain context, safety has been also referenced as *persistence* [27,79]. Roughly, safety means that the protocol is correct, thus all honest parties agree on the same operation. A persistent blockchain is not subject to *soft forks*, i.e. a split of the ledger across the nodes, and transactions cannot be cancelled or reversed [9]. Liveness indicates that the execution of a protocol correctly terminates. It embraces two dependability attributes, availability and reliability [5,80]. The former indicates the capability of a system to be accessible without interruptions, the latter states that the system executes operations without errors or faults. In a blockchain, availability is guaranteed if the infrastructure is constantly accessible and will be evaluated in the next section. Conversely, reliability ensures the correct processing of transactions and blocks [7]. We consider reliability as a liveness attribute. The properties are defined as follows:

- (1) *Persistence (safety attribute)*: If an honest node appends a block b to its local copy of the blockchain at height h , then any honest node will add b at height h locally.
- (2) *Liveness (reliability attribute)*: If a transaction is submitted to an honest node, then the transaction will be eventually added to the blockchain of all honest nodes within a time bound t (called *confirmation time*).
- (3) *Validator fairness*: consensus is fair if the likelihood of being selected as block proposer is equal for any honest validator in v blue.

Table 1 provides an evaluation of the above properties for blockchain consensus protocols. It also offers a summary of each protocol's resilience to Byzantine faults, specifically, the trust assumptions required for ensuring persistence and liveness, which define the *BFT threshold*. The table indicates with a checkmark (\checkmark) when a property is satisfied, while a cross mark (\times) for the opposite. Additionally, we employ the term '*eventual*' to denote properties that are contingent upon protocol-specific conditions.

Table 1. Dependability evaluation of blockchain consensus protocols in partially synchronous network with a minority of Byzantine faults.

	PoW	EPOS	APoS	SPoS	PBFT	AuRa	Clique
Persistence (safety)	eventual	eventual	✓	eventual	✓	✗	eventual
Liveness (reliability)	✓	✓	eventual	✓	eventual	✗	✓
Validator	hardware	stake	stake	stake	✓	✓	✓
Fairness	dependent	dependent	dependent	dependent			
BFT Threshold	$\geq 1/2$ hash power	$\geq 2/3$ stake	$\geq 3/4$ stake	$\geq 2/3$ stake	$\geq 2/3$ validators	✗	$\geq 2/3$ validators

Firstly, we discuss the results from Table 1. The probabilistic leader election adopted by PoW, EPOS, and SPoS protocols favours liveness over persistence: until there are validators with active hashing or staking power, the protocols can randomly select block proposers to update the ledger. However, this approach may lead to persistence issues having simultaneous block proposers. In this scenario, the blockchain may fork. Forks-prone systems typically employ mechanisms to ultimately converge on the same ledger after a certain confirmation time [41,42,81,82]. For this reason, we classify the persistence only as ‘eventual’ [83]. Differently, the APoS has been proven to be fork-less, with a certain mathematical probability [13]. APoS blocks get finalised, i.e. cannot be reverted from the ledger, in seconds. APoS prioritises persistence over liveness, even when facing adverse network conditions. In the event of malicious behaviours or network issues, the protocol may experience stalls. However, there is a recovery mechanism in place to eventually restore liveness. Therefore, we categorise APoS’s liveness as ‘eventual’. PoS protocols provide persistence and liveness until a majority of the stake remains honest [84]. Validators with more stake can influence block production. This behaviour is reflected in Table 1 having the validator fairness property classified as *stake dependent* for such protocols. Differently in PoW, validator fairness is said *hardware dependent*. Validators with outstanding computational power will have more chances to produce blocks and influence the network.

Shifting our focus to BFT-based protocols, we examine the widely recognised PBFT, as well as BFT-hybrid PoAs like AuRa and Clique. PBFT has been shown to prioritise persistence over liveness within partially synchronous networks. Liveness is eventually guaranteed as long as the majority of nodes are honest [53]. In contrast, PoA emphasises liveness over persistence. PoA protocols enable block processing even in adversarial conditions, leading to a forks-prone system [30]. Clique achieves persistence through the GHOST protocol, which is a dynamic chain-selection mechanism. This allows the network to eventually converge on the same fork [82]. The GHOST protocol provides ‘eventual’ persistence to Clique. AuRa relies on node synchronisation to create blocks, so it cannot guarantee persistence and liveness in a partially synchronous network, as shown in [30]. Both PoA protocols offer strong fairness to validators, who are elected deterministically in a *round-robin-based* leader selection process.

The *BFT threshold* indicates the protocol’s BFT resilience and helps identify the level of effort required by a Byzantine attacker to compromise the network. In PoW, the BFT threshold assumes that at least $\geq 1/2$ of hash power remains honest. With this assumption, PoW can eventually resolve forks using the longest chain rule [9,81]. On the other hand, in PoS systems, the BFT is tied to the staking power of validators. In fork-prone PoS systems like EPOS and SPoS, an attacker would need to compromise the fork resolution mechanism to disrupt persistence. For EPOS and SPoS, the BFT threshold is fixed at $\geq 2/3$ of the total stake being honest. In these cases, the Casper FFG and TowerBFT protocols can effectively resolve forks and achieve block finalisation. In contrast, Algorand’s APoS takes a forkless approach with instant finalisation. This is accomplished through a PBFT-like three-phase block confirmation, assuming $\geq 3/4$ of the total stake is honest [13]. In private settings, PBFT has been proven to be resilient under the assumption of $v \geq 3f + 1$ nodes, i.e. a supermajority of validators is honest. Therefore, the BFT threshold for PBFT-based blockchains must be set at $\geq 2/3$ of honest validators. For PoA protocols, AuRa lacks fault tolerance in a partially synchronous model. Consequently, even a single malicious node can cause significant disruptions in the network. Clique, proceeds under the

assumption that at least $\geq 2/3$ of validators remain honest. In this case, at least one validator can be elected as block proposer [30].

6.3. Network infrastructure

The network infrastructure of a blockchain is crucial for the evaluation of its security and dependability guarantees and their tradeoffs. For instance, public-permissionless blockchains provide strong data integrity guarantees but lack confidentiality, whereas private-permissioned blockchains ensure better privacy at the cost of decentralisation and thus data tampering resilience [1]. In this section, we consider the public-permissionless blockchains of Bitcoin and Monero with their PoW consensus, Ethereum with EPoS, Algorand with APoS, and Solana with SPoS; the private-permissioned blockchains Fabric with PBFT, and the two Ethereum-private networks Parity with AuRa PoA, and Geth with Clique PoA.

We provide six definitions. Firstly, we consider the attributes of *confidentiality*, *integrity*, and *availability* [5]. Then, we provide attributes to evaluate the access rules and control of the network infrastructure, such as *accountability* and *authorisation*. Finally, we delineate the *transaction (txn) fairness* to quantify the fair execution of transactions. Hence, our proposed attributes are:

- (1) *Confidentiality*: the possibility to keep some transactions confidential; absence of unauthorised leaking of sensitive information owned by one or more nodes;
- (2) *Integrity*: absence of improper alterations of the blockchain data from unauthorised users;
- (3) *Availability*: the system's capability to be accessible without interruptions;
- (4) *Accountability*: the system's capability to trace back the operations and the behaviour of a certain user identity/physical entity;
- (5) *Authorisation*: the system's capability to define access rights and privileges for resources, as well as to establish permission roles for participants;
- (6) *Txn fairness*: the willingness of the system to democratically accept transactions from any client without preferences or priority mechanisms.

Table 2, shows how the analysed blockchains meet the six attributes presented above. Following the discussion of the table.

6.3.1. Confidentiality, integrity, and availability

Public and permissionless blockchains, like Bitcoin, Monero, Ethereum, Algorand, and Solana, allow users to join the network without requiring permission. Anyone can set up a node and become a validator of the blockchain. Furthermore, all data stored is openly accessible to the public. These networks typically exhibit a high degree of decentralisation, boasted by a vast network of interconnected nodes [74,85]. The advantages of such a replicated infrastructure are twofold: high availability and robust data integrity. Being the nodes distributed, clients can consistently query an available node to access blockchain data and services [86]. This statement is valid for blockchains like Bitcoin, Monero, Ethereum, and Algorand, which never experienced outages. However, it is not valid for Solana, which faced several outages in the past year due to Distributed Denial of Service (DDoS) attacks [3]. The list of

Table 2. Security evaluation of blockchain platforms in partially synchronous network with Byzantine faults.

	Bitcoin	Monero	Ethereum	Algorand	Solana	Fabric	Parity	Geth
confidentiality	✗	✓	✗	✗	✗	✓	✓	✓
integrity	eventual	eventual	eventual	✓	eventual	✓	✗	eventual
availability	✓	✓	✓	✓	✗	eventual	eventual	eventual
accountability	partial	✗	partial	partial	partial	✓	✓	✓
authorisation	✗	✗	✗	✗	✗	✓	✓	✓
txn fairness	✗	✗	✗	partial	✗	✓	✓	✓

Solana's validators is publicly accessible and the leader election is deterministic [56,57]. This exposed the validators (and thus the whole system) to DDoS attacks, that resulted in network disruptions. Drifting to data integrity, public-permissionless blockchains are strong tamper-resistant systems thanks to their large and decentralised validator set (v). The finalisation process of transactions involves several parties, and it is nearly impossible to compromise all of them (or a relevant majority) [1]. Nevertheless, as previously noted the consensus protocols of Bitcoin, Monero, Ethereum, and Solana are prone to forks. This results in 'eventual' integrity guarantees. In contrast, Algorand provides a robust integrity mechanism, with transactions being finalised in a matter of seconds. Despite strong availability, the full replication of the blockchain in the Bitcoin, Ethereum, Algorand, and Solana platforms intentionally leads to a lack of confidentiality due to the public nature of the information stored on these ledgers [87]. Conversely, Monero by design preserves users' privacy through its *CryptoNote* protocol with RingSignature, RingCT, and Stealth Address. Conversely, private and permissioned blockchains offer a more controlled infrastructure for identifying and authenticating nodes. These networks trade decentralisation for trust, as highlighted in [1]. Trusted authorities establish an authentication policy for nodes, which governs the entire network's topology [60]. In private blockchains, like Fabric or Ethereum-private, the availability is directly influenced by node distribution and the SLAs² provided by the node runners. Consequently, we classify Fabric, Parity, and Geth availability as 'eventual'. On the other hand, ensuring integrity requires different infrastructural considerations. Fabric was designed as a secure and reliable platform for building private networks [76]. As soon as the BFT threshold trust assumptions are met, Fabric provides strong integrity guarantees. Differently from Fabric, Ethereum-private settings re-use Ethereum's public blockchain software, adapted to permissioned environments. In this context, hidden vulnerabilities that might be mitigated in a highly decentralised setup may turn into serious issues in private networks. In De Angelis [46], it has been demonstrated that Parity cannot guarantee integrity to a private-permissioned blockchain running in a partially synchronous network. Additionally, Geth has exhibited several vulnerabilities, as highlighted in [46,88,89], which could potentially compromise the blockchain if not adequately addressed. Consequently, we classify the integrity attribute of an Ethereum-private network built on Geth as 'eventual.' Confidentiality in Hyperledger Fabric can be guaranteed through the use of channels, i.e. private ledgers with a subset of nodes that can interact with it. Ethereum-private systems running with Parity and Geth instead do not split the ledger in channels, but given their permissioned nature, can offer confidentiality by design. Moreover, in this system there exist approaches that allow the execution of private transactions adopting the ConsenSys Tessera framework [90].

6.3.2. Accountability and authorisation

Private-permissioned blockchains like Hyperledger Fabric and the Ethereum-private can enforce the so-called profiling properties of accountability and authorisation. Accountability is achieved by tracing the interaction of nodes with the blockchain [75]. Authorisation is guaranteed by managing the permission of each node. Public permissionless blockchains – Bitcoin, Ethereum, Algorand, and Solana – have instead pseudo-anonymous identities [87] and users are not authenticated. However, although actions cannot be associated with specific end-users, it is possible to analyse the behaviour of blockchain accounts. Therefore, the public, permissionless nature of these blockchains ensures that anyone can access the history of transactions. The decentralised nature of these systems provides security guarantees that enable data integrity, and immutable audit trails validation [91,92]. We deduce that permissionless blockchain offers *partial* accountability [93,94]. Monero, on the other hand, does not provide accountability due to its strong privacy-preserving model. Finally, being these systems public and decentralised, authorisation is not provided.

6.3.3. Transaction fairness

Private-permissioned blockchains benefit from fairness guarantees. Each client's transactions are processed by validators without any preference or priority. Conversely, the execution of transactions in permissionless blockchains is costly (either hardware or staking), thus making incentive mechanisms

necessary. Low-rewards transactions may be stalled forever waiting to be processed [86]. Incentive mechanisms for permissionless blockchains, like Bitcoin, Monero, and Ethereum, lead therefore to a lack of transaction fairness. Differently, in the Algorand blockchain, every transaction counts the same, and there is no such mechanism. Everything in Algorand is handled by PPoS cryptography and the computation of VRFs. This allows Algorand to have very low transaction fees, which are thus distributed to rewards accounts for the users, and to ensure client fairness. However, in the case of network congestions, Algorand's transaction processing switches to a fee-based system, which might affect fairness as shown in [95]. For this reason, Algorand's transaction fairness is classified as 'partial'. Solana provides very cheap transactions, but transaction fairness can be tampered with by malicious validators in favour of the chosen transaction.

6.4. Smart contract applications

In Section 5 we presented a list of 15 well-known vulnerabilities exploited on the EVM. In this section, we evaluate those vulnerabilities with respect to the other smart contract platforms considered in this work, such as Algorand, Solana, and Hyperledger Fabric. The analysis excludes stateless, script-based, execution environments like Bitcoin and Monero [96,97], because not comparable with the EVM. Table 3 provides an evaluation of the 15 vulnerabilities. The table shows the impact that each vulnerability has on the security and dependability attributes. Thus, it shows with a checkmark (✓) when a blockchain offers native resistance or mitigation for the respective vulnerability.

We observe *reentrancy* (V_1) and *overflow/underflow* (V_2) first. Confidentiality and integrity are compromised due to unexpected and malicious operations. Authorisation is also compromised due to unauthorised withdrawals and unexpected interruptions [66]. Algorand provides native mitigation for both issues: contract-to-contract calls can be processed in one direction only (no reentrancy calls) and the TEAL low-level programming language natively copes with underflows and overflows. Solana partially addressed reentrancy limiting cross-programme invocations to 4-level depth calls. Conversely, Solana does not provide native prevention for overflows but offers secure and controlled mathematical functions; instead of $+$, $-$, $*$ and $/$, developers can use `checked_add`, `checked_sub`, `checked_mul` and `checked_div` opcodes. Hyperledger Fabric suffers reentrancy since 'chaincode-to-chaincode' calls are allowed with no limitations. Fabric mitigates such issues through timeouts (reentrancy has a limited impact on private settings since no cryptocurrencies are involved). Similarly to Algorand, the native smart contract programming language of Fabric (i.e. Golang) makes us of the native under/overflow management or common libraries such as *overflow*.

Table 3. Security impact of smart contract issues and native resistance/mitigation. The table shows the attributes: (C) confidentiality, (I) integrity, (A) availability, (ACC) accountability, and (AUTH) authorisation.

Issue ID	Security Impact	Native Resistance/Mitigation			
		Ethereum	Algorand	Solana	Fabric
V_1 : <i>Reentrancy</i>	C, I, AUTH		✓	✓	
V_2 : <i>Overflow/Underflow</i>	C, I, AUTH		✓		✓
V_3 : <i>Frozen Token</i>	A, AUTH			✓	✓
V_4 : <i>DoS – Revert</i>	A	✓		✓	
V_5 : <i>DoS – Gas Limit</i>	A				
V_6 : <i>Signature</i>	C, I, AUTH				✓
V_7 : <i>Randomness</i>	I, AUTH		✓		
V_8 : <i>Block Timestamp</i>	I, A			✓	
V_9 : <i>Front Running</i>	I, A, ACC				✓
V_{10} : <i>Opcodes</i>	A		✓	✓	✓
V_{11} : <i>Orphan Address</i>	A, AUTH				✓
V_{12} : <i>Short Address</i>	I, AUTH		✓	✓	✓
V_{13} : <i>Erroneous Visibility</i>	C, I, AUTH	✓		✓	✓
V_{14} : <i>Unprotected Suicide</i>	all				
V_{15} : <i>Unrequested Token</i>	AUTH		✓		✓

The *frozen token* (V_3) impacts availability and authorisation due to the impossibility of token recovery. Similarly to Ethereum, Algorand cannot offer native resistance and it is vulnerable to this issue. Solana adds innovation at the infrastructure level to mitigate funds locked in smart contracts, through the introduction of a *recovery address*, i.e. default destination addresses for smart contract funds distributed in case of incidents. Fabric is not vulnerable since no cryptocurrencies are involved. A similar security impact is caused by the *orphan address* V_{11} . Algorand and Solana do not offer native resistance. The only effective prevention method, at the time of writing, is to manually assure the correctness of the recipient's address [62]. Hyperledger nodes and identities are instead authenticated, thus there is no risk of orphan addresses.

Vulnerabilities that caused *DoS* have seriously affected the applications' availability on Ethereum smart contracts [62]. The *DoS - Revert* (V_4) is discentivised on Ethereum through rewards. Algorand cannot rely on such mitigation since no reward fees exist. Differently, Solana offers configurable rebroadcasting rules to ensure the correct processing of transactions, even in case of unexpected reverts (V_4). Fabric, finally, does not have solutions to avoid it, due to the absence of cryptocurrencies. Similarly to Ethereum, all other platforms may suffer from *DoS - GasLimit* (V_5). To avoid infinite loops, Algorand has a limited budget of computable operations (opcodes) per execution. Similarly, Solana provides a limited number of computing steps on a per-transaction basis; in both cases, the execution is interrupted once all computing units are spent. Hyperledger Fabric does not provide computing bounds but adopts timeouts. The computation constraints in place for Algorand and Solana also ensure protection against the vulnerabilities V_{10} , and V_{12} . *Under-priced opcodes* (V_{10}), which affects availability, is mitigated on both platforms thanks to their rational computational budgets assigned to complex on-chain operations (e.g. on-chain cryptographic operations). Hyperledger Fabric remains unaffected due to the inherent costless computation nature of private blockchains. Algorand and Solana also offer prevention by design for *short Address* (V_{12}) by triggering transaction failure in case of unexpected addresses, while Fabric is not affected thanks to node authentication.

Confidentiality, integrity, and authorisation attributes get compromised in case of V_6 and V_{13} . Public-permissionless blockchains are vulnerable to these issues due to their public nature. However, both Ethereum and Solana provide mitigation offering granular function visibility in smart contracts with private methods. Conversely, the private-permissioned setting of Fabric allows the adoption of privacy-protection methods like Trusted Execution Environment with Intel SGX and channels [98].

Among the most dangerous vulnerabilities are those that lead to integrity breaches. For example, the *randomness* (V_7) and *block timestamp* (V_8) vulnerabilities, if exploited, can potentially break the entire business logic of a decentralised application that relies on random numbers or a notion of time. Algorand is the only platform offering on-chain, trustless random beacons based on VRFs. Solana, instead ensures protection against timestamp manipulation (V_8) by replacing block timestamps with verifiable PoH proofs. Applications correctness is also compromised in the event of *front running* (V_8). Malicious actors take advantage of this vulnerability to manipulate the execution order prioritising more convenient transactions. It breaks integrity, availability, and accountability with unexpected execution flows. Algorand provides partial protection since transactions cannot be prioritised over others with fees. However, in the presence of network congestion, the Algorand protocol switches to a dynamic fee model, which can offer front-running possibilities [95]. Similarly, Solana provides a fee-based transaction priority mechanism. Fabric does not suffer from this issue due to the absence of cryptocurrencies. A unique anti-spam protection is given on the Algorand AVM to mitigate authorisation flaws caused by *unrequested token* V_{15} issue. On Algorand, tokens cannot be issued to users without their approval (*optin*). Native protection is also ensured on Fabric since no cryptocurrencies are involved.

Finally, *unprotected suicide* (V_{14}) is the most dangerous vulnerability from the list. It compromises all the security and dependability attributes. The Ethereum *suicide/self-destruct*, Algorand *Delete*, and Solana *Programme Close* actions lead to the irreversible destruction of a smart contract. This vulnerability can affect any platform if not properly managed by the smart contract logic.

7. Conclusion

In this paper, we analysed the security and dependability aspects of modern blockchain systems in a partially synchronous network with Byzantine faults. We first started by introducing eight well-known blockchains, i.e. Bitcoin, Monero, Ethereum, Algorand, Solana, Hyperledger Fabric, Parity, and Geth. Then, we introduced their underlying consensus protocols, such as PoW, three different versions of PoS, PBFT, and two PoA algorithms. As a first contribution the security and dependability attributes relevant to blockchains are presented. We provided a classification of the most known attributes with respect to blockchains. Then, we analysed how the chosen blockchain platforms can guarantee those attributes over three architectural layers, i.e. consensus protocol, network infrastructure, and smart contract applications. The analysis firstly focussed on the consensus protocols, and their assurance in terms of persistency, liveness, and validator fairness. For each protocol, we also discussed the BFT threshold, i.e. the resilience index to Byzantine faults. Afterwards, the analysis moved to network infrastructures. We introduced six dependability and security attributes, of which accounting, authorisation, and transaction fairness are of novel contribution. We evaluated those attributes against the eight blockchains under study. Lastly, we listed well-known Ethereum smart contract vulnerabilities, and we outlined how other smart contract platforms are susceptible/resistant and whether they provide native mitigations. From this study emerged that permissioned blockchains like Hyperledger Fabric provide a trusted and more controlled environment that guarantees better confidentiality, accountability, and authentication. However, these platforms require strong assumptions on the network reliability (e.g. nodes' SLAs) in order to offer simultaneous integrity and availability. Conversely, public-permissionless platforms offer better integrity and availability guarantees thanks to higher decentralisation at the cost of accounting, authentication, confidentiality, and client fairness. Among the permissionless blockchains, we outlined how different designs provide divergent integrity and availability guarantees. For example, fork-prone systems like Bitcoin, Monero, and Ethereum favour availability to integrity. Conversely, Algorand prefers strong integrity with a fork-less model, preserving the integrity of the ledger even in case of adverse network conditions. Finally, we demonstrated how Solana's deterministic leader election impacts security and dependability due to eventual DoS attacks against the validators set. From this analysis, we observed that other smart contract platforms provide generally better resilience and mitigation to smart contract vulnerabilities famous on the Ethereum EVM.

As a future direction, we aim to expand the analysis to other blockchain platforms and improve the analysis to other smart contract issues. Thus we aim to elevate this study with a quantitative evaluation of our attributes and extend the measurement of performance alongside security and dependability.

Notes

1. The Ethereum's cryptocurrency.
2. SLA – Service Level Agreement.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

- [1] Troncoso C, Isaakidis M, Danezis G, et al. Systematizing decentralization and privacy: lessons from 15 years of research and deployments. *Proc Priv Enh Technol*. 2017;2017:404–426.
- [2] CertiK. The state of defi security; 2021. Available from: <https://certik-2.hubspotpagebuilder.com/the-state-of-defi-security-2021>
- [3] Solana 9-14 network outage initial overview; 2021. [accessed 2023 Sept 10]. Available from: <https://solana.com/news/9-14-network-outage-initial-overview>
- [4] Wiki B. Bitcoin value overflow incident; 2016. [accessed 2023 Sept 20]. Available from: https://en.bitcoin.it/wiki/Value_overflow_incident

- [5] Avizienis A, Laprie JC, Randell B, et al. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans Dep Secur Comput.* 2004;1(1):11–33. doi: [10.1109/TDSC.2004.2](https://doi.org/10.1109/TDSC.2004.2)
- [6] Avizienis A, Laprie JC, Randell B. Fundamental concepts of dependability; 2001.
- [7] Cachin C, Vukolic M. Blockchain consensus protocols in the wild; 2017. CoRR. abs/1707.01873.
- [8] Shehar B, Alberto S, Mustafa AB, et al. Sok: Consensus in the age of blockchains. In: Proceedings of the 1st ACM Conference on Advances in Financial Technologies; 2019.
- [9] Vukolić M. The quest for scalable blockchain fabric: proof-of-work vs. bft replication. In: Camenisch J, Kesdoğan D, editors. Open problems in network security; 2016.
- [10] Nakamoto S. Bitcoin: a peer-to-peer electronic cash system; 2008. Available from: <https://bitcoin.org/bitcoin.pdf>
- [11] Project M. Monero; 2014. Available from: <https://www.getmonero.org/>
- [12] Ethereum. Vision of ethereum2; 2020. Available from: <https://ethereum.org/en/upgrades/vision/>
- [13] Gilad Y, Hemo R, Micali S, et al. Algorand: scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles; 2017.
- [14] Yakovenko A. Solana: a new architecture for a high performance blockchain v0.8.13; 2017. Available from: <https://solana.com/solana-whitepaper.pdf>
- [15] Hyperledger fabric; 2015. Available from: <https://www.hyperledger.org/use/fabric>
- [16] Geth. Ethereum; 2016. Available from: <https://geth.ethereum.org/docs/interface/private-network>
- [17] Mense A, Flatscher M. Security vulnerabilities in ethereum smart contracts. In: Proceedings of the 20th IWBAS; 2018.
- [18] Atzei N, Bartoletti M, Cimoli T. A survey of attacks on ethereum smart contracts (sok). In: POST – 6th International Conference, Proceedings; Lecture Notes in Computer Science; Vol. 10204. Springer; 2017. p. 164–186. Springer-Verlag, Berlin, Heidelberg.
- [19] Samreen NF, Alalfi MH. A survey of security vulnerabilities in ethereum smart contracts; 2021. CoRR. abs/2105.06974. Available from: <https://arxiv.org/abs/2105.06974>
- [20] Kannengießer N, Lins S, Dehling T, et al. Trade-offs between distributed ledger technology characteristics. *ACM Comput Surv.* 2021;53(2):1–37. doi: [10.1145/3379463](https://doi.org/10.1145/3379463)
- [21] Xiao Y, Zhang N, Lou W, et al. A survey of distributed consensus protocols for blockchain networks. *IEEE Commun Surv Tutor.* 2020;22(2):1432–1465. doi: [10.1109/COMST.9739](https://doi.org/10.1109/COMST.9739)
- [22] Mingxiao D, Xiaofeng M, Zhe Z, et al. A review on consensus algorithm of blockchain; 2017.
- [23] Sankar LS, Sindhu M, Sethumadhavan M. Survey of consensus protocols on blockchain applications. In: ICACCS. IEEE; 2017. p. 1–5. Coimbatore, India.
- [24] Kiffer L, Rajaraman R, Shelat A. A better method to analyze blockchain consistency. In: Lie D, Mannan M, Backes M, et al., editors. Proceedings of the 2018 SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, Oct 15–19. ACM; 2018. p. 729–744. Machinery, New York, NY, USA.
- [25] Pass R, Seeman L, Shelat A. Analysis of the blockchain protocol in asynchronous networks. In: Advances in Cryptology – 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, April 30–May 4; Lecture Notes in Computer Science. Vol. 10211; 2017. p. 643–673.
- [26] Saltini R. Correctness analysis of IBFT; 2019. CoRR. abs/1901.07160.
- [27] Saltini R. IBFT liveness analysis. In: International Conference on Blockchain, Blockchain 2019, Atlanta, July 14–17. IEEE; 2019. p. 245–252. Atlanta, GA, USA.
- [28] Amoussou-Guenou Y, Potop-Butucaru MG, et al. Correctness and fairness of tendermint-core blockchains; 2018. ArXiv abs/1805.08429.
- [29] Ekparinya P, Gramoli V, Jourjon G. The attack of the clones against proof-of-authority. In: 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, 2020 February 23–26. The Internet Society; 2020.
- [30] De Angelis S, Aniello L, Baldoni R, et al. PBFT vs proof-of-authority: applying the CAP theorem to permissioned blockchain. In: Proceedings of ITASEC; 2018.
- [31] Buterin V. Why sharding is great: demystifying the technical properties; 2021. Available from: <https://vitalik.ca/general/2021/04/07/sharding.html>
- [32] Rosenfeld M. Analysis of hashrate-based double spending; 2014.
- [33] Wu S, Chen Y, Li M, et al. Survive and thrive: a stochastic game for DDoS attacks in bitcoin mining pools. *IEEE ACM Trans Netw.* 2020;28(2):874–887. doi: [10.1109/TNET.90](https://doi.org/10.1109/TNET.90)
- [34] Van Saberhagen N. Cryptonote v 2.0; 2013.
- [35] Wood G. Ethereum: a secure decentralised generalised transaction ledger; 2014.
- [36] Ethereum. Solidity; 2015. Available from: <https://soliditylang.org>
- [37] Vyper smart-contracts programming language; 2019. Available from: <https://vyper.readthedocs.io/en/stable/>
- [38] Buterin V. Dagger: a memory-hard to compute, memory-easy to verify scrypt alternative. Tech Report, hashcash.org website; 2013.
- [39] Dryja T. Hashimoto: I/o bound proof of work; 2009.
- [40] Bez M, Fornari G, Vardanega T. The scalability challenge of ethereum: An initial quantitative analysis. In: 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE); 2019. p. 167–176.
- [41] Stewart A, Kokoris-Kogia E. Grandpa: a byzantine finality gadget; 2020.

- [42] Buterin V, Griffith V. Casper the friendly finality gadget; 2017.
- [43] Go ethereum – geth; 2016. Available from: <https://geth.ethereum.org>
- [44] Parity ethereum; 2018. Available from: <https://www.parity.io>
- [45] Rouhani S, Deters R. Performance analysis of ethereum transactions in private blockchain. In: 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS); 2017. p. 70–74.
- [46] De Angelis S. Assessing security and performance of blockchain systems and consensus protocols: taxonomies, methodologies and benchmarking procedures [dissertation]. University of Southampton; 2022.
- [47] Algorand; 2018. Available from: <https://www.algorand.com>
- [48] Bonneau J, Miller A, Clark J, et al. Sok: research perspectives and challenges for bitcoin and cryptocurrencies. In: 2015 IEEE Symposium on Security and Privacy. IEEE; 2015. p. 104–121. San Jose, CA, USA.
- [49] Gervais A, Karame GO, Wüst K, et al. On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York (NY): Association for Computing Machinery. CCS '16; 2016. p. 3–16. doi: [10.1145/2976749.2978341](https://doi.org/10.1145/2976749.2978341)
- [50] King S, Nadal S. PPCoin: peer-to-peer crypto-currency with proof-of-stake; 2012.
- [51] Ge L, Wang J, Zhang G, et al. Survey of consensus algorithms for proof of stake in blockchain. *Sec Commun Netw*. 2022;2022:Article ID 2812526.
- [52] Micali S, Rabin M, Vadhan S. Verifiable random functions. In: 40th annual symposium on foundations of computer science. IEEE; 1999. p. 120–130. New York, NY, USA.
- [53] Castro M, Liskov B. Practical byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation. Berkeley (CA): USENIX Association. OSDI '99; 1999. p. 173–186.
- [54] Boneh D, Bonneau J, Bünz B, et al. Verifiable delay functions [Cryptology eprint archive, paper 2018/601]; 2018. Available from: <https://eprint.iacr.org/2018/601>
- [55] Shoup V. Proof of history: what is it good for?; 2022. Available from: <https://www.shoup.net/papers/poh.pdf>
- [56] Solana proof of stake and proof of history primer; 2022. [accessed 2023 Sept 10]. Available from: <https://www.shinobi-systems.com/primer.html>
- [57] Solana leader rotation; 2020. [accessed 2023 Sept 10]. Available from: <https://docs.solana.com/cluster/leader-rotation>
- [58] Parity. Aura – authority round consensus protocol; 2016. Available from: <https://wiki.parity.io/Aura>
- [59] Szilágyi P. Clique – ethereum proof-of-authority consensus protocol; 2016. Available from: <https://github.com/ethereum/EIPs/issues/225>
- [60] BitFury Group, Garzik J. Public versus private blockchains part 1: permissioned blockchains. White Paper; 2015. Available from: <http://bitfury.com/content/5-white-papers-research/public-vs-private-pt1-1.pdf>
- [61] Gaetani E, Aniello L, Baldoni R, et al. Blockchain-based database to ensure data integrity in cloud computing environments. In: ITA-SEC; Vol. 1816. CEUR-WS.org; 2017.
- [62] Chen H, Pendleton M, Njilla L, et al. A survey on ethereum systems security: vulnerabilities, attacks, and defenses. *ACM Comput Surv*; 2020.
- [63] Rameder H, di Angelo M, Salzer G. Review of automated vulnerability analysis of smart contracts on ethereum. *Frontiers Blockchain*. 2022;5:Article ID 814977. doi: [10.3389/fbloc.2022.814977](https://doi.org/10.3389/fbloc.2022.814977)
- [64] Dasp – decentralized application security project top 10; 2018. [accessed 2023 Sept 20]. Available from: <https://dasp.co/>
- [65] Eea ethtrust security levels specification v1; 2022. [accessed 2023 Sept 20]. Available from: <https://entethalliance.org/specs/ethtrust-sl/>
- [66] Rodler M, Li W, Karame GO, et al. Sereum: protecting existing smart contracts against re-entrancy attacks; 2018. CoRR. abs/1812.05934.
- [67] OpenZeppelin. Solidity safemath library; 2018. Available from: <https://docs.openzeppelin.com/>
- [68] ConsenSys. Mythril; 2018. Available from: <https://github.com/ConsenSys/mythril>
- [69] Qian P, He J, Lu L, et al. Demystifying random number in ethereum smart contract: taxonomy, vulnerability identification, and attack detection. *IEEE Trans Softw Eng*. 2023;49(7):3793–3810. doi: [10.1109/TSE.2023.3271417](https://doi.org/10.1109/TSE.2023.3271417)
- [70] Breidenbach L, Cachin C, Chan B, et al. Chainlink 2.0: next steps in the evolution of decentralized oracle networks; 2021. [accessed 2023 Sept 19]. Available from: <https://research.chain.link/whitepaper-v2.pdf>
- [71] Wang Y, Chen Y, Deng S, et al. Cyclic arbitrage in decentralized exchange markets; 2021. ArXiv abs/2105.02784.
- [72] Rahimian R, Eskandari S, Clark J. Resolving the multiple withdrawal attack on erc20 tokens. In: 2019 IEEE European Symposium on Security and Privacy Workshops; 2019. p. 320–329.
- [73] Liu Y, Lu Q, Yu G, et al. Defining blockchain governance principles: a comprehensive framework. *Inf Syst*. 2022;109:Article ID 102090. doi: [10.1016/j.is.2022.102090](https://doi.org/10.1016/j.is.2022.102090)
- [74] BitFury Group, Garzik J. Public versus private blockchains part 2: permissionless blockchains. White Paper; 2015. Available from: <http://bitfury.com/content/5-white-papers-research/public-vs-private-pt2-1.pdf>
- [75] Herlihy M, Moir M. Enhancing accountability and trust in distributed ledgers; 2016. CoRR.
- [76] Graf M, Küsters R, Rausch D. Accountability in a permissioned blockchain: formal analysis of hyperledger fabric. In: 2020 IEEE European Symposium on Security and Privacy (EuroSP); 2020. p. 236–255.
- [77] Francez N. Fairness. Berlin, Heidelberg: Springer; 1986.

- [78] Dwork C, Lynch N, Stockmeyer L. Consensus in the presence of partial synchrony. *J ACM*. 1988;35(2):288–323. doi: [10.1145/42282.42283](https://doi.org/10.1145/42282.42283)
- [79] Kiayias A, Russell A, David B, et al. Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz J, Shacham H, editors. *Advances in Cryptology – CRYPTO 2017*. Cham: Springer International Publishing; 2017. p. 357–388.
- [80] Gilbert S, Lynch NA. Perspectives on the cap theorem. *Computer*. 2012;45(2):30–36. doi: [10.1109/MC.2011.389](https://doi.org/10.1109/MC.2011.389). Available from: <https://api.semanticscholar.org/CorpusID:842354>
- [81] Courtois NT. On the longest chain rule and programmed self-destruction of crypto currencies; 2014. CoRR. Available from: <http://arxiv.org/abs/1405.0534>
- [82] Sompolinsky Y, Zohar A. Secure high-rate transaction processing in bitcoin. In: Böhme R, Okamoto T, editors. *Financial Cryptography and Data Security – 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26–30, 2015, Revised Selected Papers*; (Lecture Notes in Computer Science; Vol. 8975). Springer; 2015. p. 507–527. doi: [10.1007/978-3-662-47854-7_32](https://doi.org/10.1007/978-3-662-47854-7_32). San Juan, Puerto Rico.
- [83] Vukolic M. Eventually returning to strong consistency. *IEEE Data Eng Bull*. 2016;39:39–44. Available from: <https://api.semanticscholar.org/CorpusID:12818469>
- [84] Nguyen CT, Hoang DT, Nguyen DN, et al. Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. *IEEE Access*. 2019;7:85727–85745. doi: [10.1109/Access.6287639](https://doi.org/10.1109/Access.6287639)
- [85] Neudecker T, Hartenstein H. Network layer aspects of permissionless blockchains. *IEEE Commun Surv Tutor*. 2019;21(1):838–857. doi: [10.1109/COMST.9739](https://doi.org/10.1109/COMST.9739)
- [86] Weber I, Gramoli V, Ponomarev A, et al. On availability for blockchain-based systems. In: *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. IEEE; 2017. p. 64–73. Hong Kong, Hong Kong.
- [87] Henry R, Herzberg A, Kate A. Blockchain access privacy: challenges and directions. *IEEE Security Privacy*. 2018;16(4):38–45. doi: [10.1109/MSP.2018.3111245](https://doi.org/10.1109/MSP.2018.3111245)
- [88] Chen Y, Ma F, Zhou Y, et al. Tyr: finding consensus failure bugs in blockchain system with behaviour divergent model. In: *2023 IEEE Symposium on Security and Privacy (SP)*; 2023. p. 2517–2532.
- [89] Yang Y, Kim T, Chun BG. Finding consensus bugs in ethereum via multi-transaction differential fuzzing. In: *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*; Jul. USENIX Association; 2021. p. 349–365. Online.
- [90] ConsenSys. Tesseract – privacy transaction manager; 2023. [accessed 2023 Sept 15]. Available from: <https://docs.tesseract.consenSys.net>
- [91] Ahmad A, Saad M, Njilla L, et al. Blocktrail: a scalable multichain solution for blockchain-based audit trails. In: *2019 IEEE International Conference on Communications, ICC 2019, Shanghai, China, May 20–24, 2019*. IEEE; 2019. p. 1–6. Shanghai, China.
- [92] Kalis R, Belloum A. Validating data integrity with blockchain. In: *2018 IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2018, Nicosia, Cyprus, December 10–13, 2018*. IEEE Computer Society; 2018. p. 272–277. Nicosia, Cyprus.
- [93] Karamé GO, Andrólaki E, Roeschlin M, et al. Misbehavior in bitcoin: a study of double-spending and accountability. *ACM Trans Inf Syst Secur*. 2015;18(1):2:1–2:32. doi: [10.1145/2732196](https://doi.org/10.1145/2732196)
- [94] Möser M. Anonymity of bitcoin transactions an analysis of mixing services; 2013.
- [95] Öz B, Gebele J, Rezabek F, et al. A first study of MEV on an up-and-coming blockchain: algorand; 2023.
- [96] Bistarelli S, Mercanti I, Santini F. An analysis of non-standard transactions. *Frontiers Blockchain*. 2019;2:7. doi: [10.3389/fbloc.2019.00007](https://doi.org/10.3389/fbloc.2019.00007)
- [97] Bistarelli S, Bracciali A, Klomp R, et al. Towards automated verification of bitcoin-based decentralised applications. In: *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*. New York (NY): Association for Computing Machinery; 2023. p. 262–269; SAC '23.
- [98] Gao W, Hei X, Wang Y. The data privacy protection method for hyperledger fabric based on trustzone. *Mathematics*. 2023;11(6):1357. doi: [10.3390/math11061357](https://doi.org/10.3390/math11061357). Available from: <https://www.mdpi.com/2227-7390/11/6/1357>