The Security Reference Architecture for Blockchains: Towards a Standardized Model for Studying Vulnerabilities, Threats, and Defenses

Ivan Homoliak[*][†] Sarad Venugopalan[*] Qingze Hum[*] Daniel Reijsbergen[*]
Richard Schumi[*] Pawel Szalachowski[*]
[*]Singapore University of Technology and Design
[†]Brno University of Technology

***Abstract*—Due to their specific features, such as decentralization and immutability, blockchains have become popular in recent years. Blockchains are full-stack distributed systems in terms of realization, where security is a critical factor for their success. However, despite increasing popularity and adoption, there is a lack of standardized models to study security threats related to blockchains in a similar fashion as was done, e.g., in the area of cloud computing [236], [384]. To fill this gap, the main focus of our work is to systematize the knowledge about security and privacy aspects of blockchains, and thus contribute to the standardization of this domain.**

To this end, we propose the security reference architecture for blockchains, which utilizes a stacked model (similar to the ISO/OSI) that demonstrates the nature and hierarchy of various security and privacy threats. The model contains four layers: (1) the network layer, (2) the consensus layer, (3) the replicated state machine layer, and (4) the application layer. At each of these layers, we identify known security threats, their origin as well as mitigation techniques or countermeasures. Although a similar model has already been used in previous work to serve as a general outline of the blockchain infrastructure, we adapt it for the purpose of studying security threats in this domain. Further, we propose a blockchain-specific version of the threat-risk assessment standard ISO/IEC 15408 by embedding the stacked model into this standard. Finally, following our stacked model and its categorization, we provide an extensive survey of blockchain-oriented and related research as well as its applications.

***Index Terms*—Reference architecture, blockchains, distributed ledgers, security, privacy, vulnerabilities, threats, ISO/IEC 15408.**

I. Introduction

The popularity of blockchain systems has rapidly increased in recent years, mainly due to the decentralization of control that they aim to provide. Blockchains are full-stack distributed systems in which multiple layers, (sub)systems, and dynamics interact together. Hence, they should leverage a secure and resilient networking architecture, a robust consensus protocol, and a safe environment for building higher-level applications. Although most of the deployed blockchains need better scalability and well-aligned incentives to unleash their full potential, their security is undoubtedly a critical factor for their success. As these systems are actively being developed and deployed, it is often challenging to understand how secure they are, or what security implications are introduced by some specific components they consist of. Moreover, due to their complexity and novelty (e.g., built-in protocol incentives), their security assessment and analysis requires a more holistic view than in the case of traditional distributed systems.

Although some standardization efforts have already been undertaken, they are either specific to a particular platform [130] or still under development [195], [194]. Hence, there is a lack of platform-agnostic standards in blockchain implementation, interoperability, services, and applications, as well as the analysis of its security threats [160], [26]. Although all of these areas are challenging, and it might take years until they are standardized and agreed across a diverse spectrum of stakeholders, in this work, we aim to contribute to the standardization of security threat analysis. We believe that it is critical to provide blockchain stakeholders (developers, users, standardization bodies, regulators, etc.) with a comprehensive systematization of knowledge about the security and privacy aspects of today's blockchain systems.

In this work, we aim to achieve this goal, with a particular focus on system design and architectural aspects. We do not limit our work to an enumeration of security issues, but additionally, discuss the origins of those issues while listing possible countermeasures and mitigation techniques together with their potential implications. As our main contribution, we propose the *security reference architecture for blockchains*, which is based on models that demonstrate the stacked hierarchy of different threat categories (similar to the ISO/OSI hierarchy [404]). As our next contribution, we enrich the threat-risk assessment standard ISO/IEC 15408 [105] to fit the blockchain infrastructure. We achieve this by embedding the stacked model into this standard.

This paper is based on our recent work outlining the security reference architecture [189]. We substantially modify and extend it by the following:

- We provide a theoretical background related to the security reference architecture and the environment of blockchains, their types, failure models, consensus protocols, design goals, and means to achieve these goals.
- For each layer, we model particular attacks or their categories through vulnerability/threat/defense graphs, while we provide reasoning about several important relations and causalities in these graphs.
- We modify and significantly extend the application layer, where we propose a novel functionality-oriented catego-

rization, as opposed to the application-domain-oriented categorizations presented in related work [88], [399]. On a higher level, we split the application category into two groups: ecosystem applications and higher-level applications.
- We extend and revise the consensus layer by mining-pool-specific attacks, time-spoofing attacks, and we provide a more fine-grained categorization.
- For each layer, we present an incident table that lists and briefly describes examples of attacks that have occurred worldwide: either caused by malicious parties or by researchers who demonstrated their practical feasibility.

The rest of the paper is organized as follows: We describe the scope of our research and the employed methodology in Section II. In Section III we summarize the background on blockchain systems. Next, in Section IV we introduce the security reference architecture whose layers are discussed in the follow-up sections. In detail, Section V deals with the security and privacy of the network layer of our model, Section VI focuses on the consensus layer, Section VII overviews the replicated state machine layer, and Section VIII with Section IX describe applications built on top of the blockchain. We discuss the limitations of our work in Section X and we compare our research to related work in Section XI. Finally, we conclude the paper in Section XII.

II. Methodology & Scope

In contrast to conventional survey approaches, such as grounded theory for rigorous literature review [380], we do not sample included research from existing databases queried with specific terms. Instead, we study and analyze security-oriented literature for vulnerabilities and threats related to the blockchain infrastructure. The literature that we select as the input covers mainly existing blockchain-oriented surveys as well as the best conferences and journals in security and distributed computing. However, we also include other materials published in preprints, whitepapers, products' documentations and descriptions, and forums, which also represent related and valuable information.

We aim to consolidate the literature, categorize found vulnerabilities and threats according to their origin, and in the result, we create four main categories (also referred to as layers). At the level of particular main categories, we apply sub-categorization that is based on the existing knowledge and operation principles specific to such subcategories, especially concerning the security implications. If some subcategories impose equivalent security implications, we merge them into a single subcategory. Further, our next aim is to indicate and explain the co-occurrences or relations of multiple threats, either at the same main category or across more categories.

The scope of our work mainly covers aspects related to the blockchain core elements, while we mention common operational security issues (e.g., key management in the blockchain ecosystem) and countermeasures only tangentially if required. Similarly, we do not pursue threats that emerge outside of the blockchain infrastructure and outside of the extra infrastructure required for certain blockchain-based applications. Out-of-scope examples are remote exploitation of arbitrary devices (e.g., covert mining/cryptojacking [354]) and issues related to using blockchain explorers (e.g., spoofing attacks, availability issues). Examples of vulnerabilities that we assume only tangentially are semantic bugs and programming errors in the infrastructure of the blockchains – we assume that core blockchain infrastructure is implemented correctly, uses secure cryptographic primitives, and provides expected functionality (see further discussion in Section X).

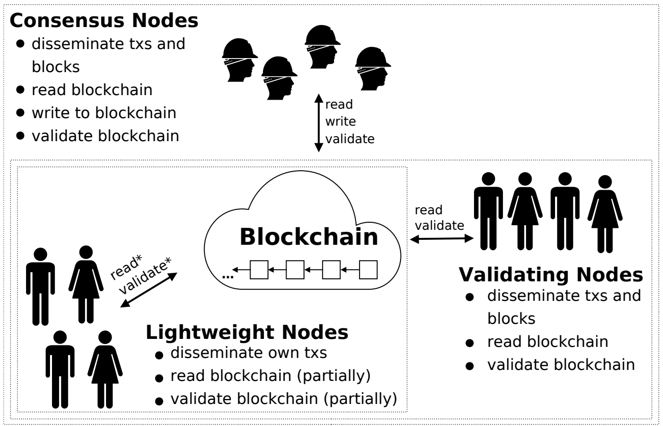

Figure 1: Involved parties with their interactions and hierarchy.

III. Blockchains at a Glance

The blockchain is a data structure representing an append-only distributed ledger that consists of entries (a.k.a., transactions) aggregated within ordered blocks. The order of the blocks is agreed upon by mutually untrusting participants running a consensus protocol – these participants are also referred to as nodes. The blockchain is resistant against modifications by design, since blocks are linked using a cryptographic hash function, and each new block has to be agreed upon by nodes running a consensus protocol. A transaction is an elementary data entry that may contain arbitrary data, e.g., an order to transfer native cryptocurrency (i.e., crypto-tokens), a piece of application code (i.e., smart contract), the execution orders of such application code, etc. Transactions sent to a blockchain are validated by all nodes that maintain a replicated state of the blockchain.

A. Involved Parties

Blockchains usually involve three native types of parties that can be organized into a hierarchy, according to the actions that they perform (see Figure 1):

(1) Consensus nodes actively participate in the underlying consensus protocol. These nodes can read the blockchain and write to it by appending new transactions. Additionally, they can validate the blockchain and thus check whether writes of other consensus nodes are correct in the sense that they respect a specified logic. By a combination of writing and validation capabilities, consensus nodes can prevent malicious behavior (e.g., by not appending invalid transactions, or not following an incorrect blockchain view). These nodes disseminate transactions to be appended within a block to the blockchain. In the context of Proof-of-Resource protocols (see Section VI-B), these nodes are often referred to as *miners*.

(2) Validating nodes read the entire blockchain, validate it, and disseminate transactions to be appended to the blockchain. Unlike consensus nodes, validating nodes cannot write to the blockchain. Thus, they cannot prevent malicious behavior. However, since they possess copies of the entire blockchain, they can detect malicious behavior.

(3) Lightweight nodes (a.k.a., light clients, or just clients) benefit from most of the blockchain functionalities, but they are equipped only with limited information about the blockchain. These nodes read only a fragment of the blockchain (usually block headers) and validate only a small number of transactions that concern them, while they rely on consensus and validating nodes for ensuring the correctness of the blockchain. Therefore, they can detect only a limited set of attacks, usually pertaining to their own transactions.

Additional Involved Parties. Besides native types of involved parties, many applications (see Section VIII and Section IX) using or running on the blockchain introduce their own components that provide additional (centralized) services or functionalities, such as data-storing servers, hosted wallets, components that contain trusted computing equipment, etc.

B. Features of Blockchains

Blockchains were initially introduced as a means of coping with the centralization of monetary assets management, resulting in their most popular application – a decentralized cryptocurrency with native crypto-tokens. However, other blockchain applications have meanwhile started to proliferate as well, benefiting from features other than decentralization. We summarize the inherent and non-inherent features of blockchains in the following.

1) Inherent Features:

Decentralization: is achieved thanks to a distributed consensus protocol – the protocol ensures that each modification of the ledger is a result of interaction among participants. In the consensus protocol, participants are equal, i.e., no single entity is designed as an authority. An important result of decentralization is resilience to node failures.

Immutability: means that the history of the ledger cannot be easily modified – it requires a significant quorum of colluding nodes. The immutability of history is achieved by a cryptographic one-way function (i.e., a hash function) that creates integrity-preserving links between the previous record (i.e., block) and the current one. In this way, integrity-preserving chains (e.g., blockchains) or graphs (e.g., direct acyclic graphs [341], [356], [300] or trees [342]) are built in an append-only fashion.

Availability: although distributed ledgers are highly redundant in terms of data storage (i.e., full nodes store replicated data), the main advantage of such a redundancy is payed off by extremely high availability of the system. This feature may be of special interest to applications that cannot tolerate outages.

Auditability: correctness of each transaction and block recorded in the blockchain can be validated by any participating node, which is possible thanks to the publicly-known rules of a consensus protocol.

Transparency: the transactions stored in the blockchain, as well as the actions of protocol participants, are (at least partially) visible to other participants and in most cases, even to the public.

2) Non-Inherent Features:

Additionally to the inherent features, blockchains may be equipped with other features that aim to achieve extra goals. Below we list a few examples of such non-inherent features.

Energy Efficiency: running an open distributed ledger often means that scarce resources are wasted (e.g., Proof-of-Work). However, there exist consensus protocols that do not waste such resources, but instead emulate the consumption of scarce resources (i.e., proof-of-burn), or the interest rate on an investment (i.e., proof-of-stake). See examples of these protocols in Section VI-B and Section VI-D.

Scalability: describes how the consensus protocol scales when the number of participants increases. Protocols whose behavior is not negatively affected by an increasing number of participants have a high scalability.

Throughput: represents the number of transactions that can be processed per unit of time. Some consensus protocols have only a small throughput (e.g., Proof-of-Work), while others are designed with the intention to maximize throughput (e.g., Byzantine Fault Tolerant (BFT) protocols with a small number of participants). See examples of BFT protocols in Section VI-C.

Privacy & Anonymity: by design, data recorded on a blockchains is visible to all nodes or public, which may lead to privacy and anonymity issues. Therefore, multiple solutions increasing the anonymity (e.g., ring signatures [313] in Monero) and privacy (e.g., zk-SNARKs [30] in Zcash) were proposed in the context of cryptocurrencies, while other efforts have been made in privacy-preserving smart contract platforms [221], [96].

Accountability and Non-Repudiation: if blockchains or applications running on top of them are designed in such a way that identities of nodes (or application users) are known and verified, accountability and non-repudiation of actions performed can also be provided.

C. Types of Blockchains

Based on how a new node enters a consensus protocol, we distinguish the following blockchain types:

Permissionless blockchains allow anyone to join the consensus protocol without permission. Such participation can be anonymous (or pseudonymous), and these protocols are designed to run over the Internet. To prevent Sybil attacks [122], these schemes usually require consensus nodes to establish their identities by running a Proof-of-Resource scheme, while the consensus power of a node

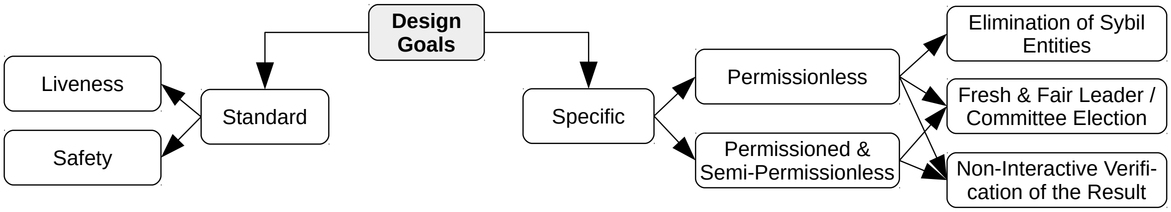
Design Goals
Standard
Specific
Liveness
Safety
Permissionless
Permissioned & Semi-Permissionless
Elimination of Sybil Entities
Fresh & Fair Leader / Committee Election
Non-Interactive Verification of the Result

Figure 2: Standard and specific design goals of consensus protocols.

is proportional to its resources invested into running the protocol.

Permissioned blockchains require a consensus node to obtain permission to join the consensus protocol. The permission is given by a centralized or federated authority(ies), while nodes usually have equal consensus power (i.e., one vote per node). These schemes can be *public* if they are accessible over the Internet or *private* when they are deployed over a restricted network.

Semi-Permissionless blockchains require each new-coming consensus node to obtain some form of permission (i.e., cryptocurrency "stake"); however, such permission can be given by any stakeholder (i.e., consensus node). These blockchains are similar to permissionless blockchains, except the consensus is based on a stake invested rather than on resources spent. The node's consensus power is proportional to the stake that it has. Similar to permissionless blockchains, these systems are usually intended to be run over the Internet. Novel and interesting aspects of permissionless and semi-permissionless blockchains are incentives and network effects that are designed to increase the protocol's security, deployability, and adoption.

D. Design Goals of Consensus Protocols

The standard design goals of distributed consensus protocols are *liveness* and *safety*. **Liveness** ensures that all transactions that are produced are eventually processed – i.e., if a transaction is received by a single honest node, it will eventually be delivered to all honest nodes. **Safety** ensures that if an honest node accepts (or rejects) a transaction, then all other honest nodes make the same decision.[1] It is worth to note that with regards to the safety, literature often use the term *finality* and *time to finality*. **Finality** represents the state of the blockchain from its genesis block up to the block B, where it can be assumed that this sequence of blocks is infeasible to overturn. To reach finality up to the block B, several successive blocks after B need to be produced and appended to the blockchain. For example, in Bitcoin [265], six such successive blocks are typically considered enough to reach finality. In contrast, BFT protocols can reach finality faster – one consecutive block is often enough. Note that the number of successive blocks required for reaching the finality is often referred to as *the number of confirmations*.

Usually, consensus protocols satisfy safety and liveness properties only under certain assumptions: typically regarding the minimal fraction of honest consensus power, or the maximal tolerated fraction of adversarial consensus power. For example, in Nakamoto consensus [265], the minimal fraction of the honest consensus power is 51%.[2] In contrast, most of the BFT protocols require this fraction to be at least 66.7% and some even more (e.g., 75% for ELASTICO [240]). We further discuss these and other consensus protocol assumption in Section VI-A.

We highlight that the type of a blockchain influences the specific design goals of its consensus protocol, which are considered by designers of blockchains on top of the standard design goals and the inherent features (see Section III-B1). In the following, we elaborate on specific design goals to particular blockchain types as well as on means to achieve design goals (see Figure 2).

Permissionless Type. One of the first design goals of permissionless protocols is to *eliminate Sybil entities* – such elimination can be done by requiring that some amount of scarce resources is spent in order to write to the blockchain, and hence no Sybil entity can participate. The next design goal of these consensus protocols is *a fresh and fair leader/committee election*, which ensures that each consensus node influences the result of a consensus protocol in a statistical ratio that is commensurate to the amount of scarce resources spent by that node. Moreover, freshness avoids prediction of the elected nodes, and thus elected nodes cannot became the subject of targeted DoS attacks. The last design goal is *non-interactive verification* of the result of the consensus – i.e., any node can verify the result without interaction.

Permissioned and Semi-Permissionless Types. These types of blockchains require *fresh and fair leader/committee election* as well as *non-interactive verification* of the result of the consensus. However, in contrast to the permissionless blockchains, they do not require a means for the elimination of Sybil entities, as permission to enter the system is given by a centralized entity (i.e., permissioned type) or any of existing consensus node (i.e., semi-permissionless type).

1) Means to Achieve Design Goals:

Simulation of VRF. To ensure a fresh and fair leader / committee election, all consensus nodes should contribute to the pseudo-randomness generation that determines the fresh result of the election. This is captured by the concept of

[1] Another definition of safety states that the interpretation of transactions satisfies linearizability [181] – i.e., decentralized execution behaves like a centralized service that executes operations atomically. However, we note that this definition applies only to blockchains that build on top of a single chain, while it does not apply to blockchains with graph structure such as DAGs.

[2] Note that even with higher fractions certain adversarial strategies such as selfish mining may become profitable (see Section VI-B).

Verifiable Random Function (VRF) [251], which ensures the unpredictability and fairness of the election process. Therefore, the leader / committee election process can be viewed as a simulation of VRF [373]. Thanks to the properties of VRF, the correctness of the election result can be verified non-interactively after the election took place.

Incentive and Rewarding Schemes. An important aspect for protocol designers is to include a rewarding/incentive scheme that motivates consensus nodes to participate honestly in the protocol. In the context of public (permissionless) cryptocurrencies that embed their native crypto-tokens, this is achieved by introducing block creation rewards as well as transaction fees, and optionally penalties for misbehavior. Transaction fees and block creation rewards are attributed to the consensus node(s) that create a valid block (e.g., [265]), although alternative incentive schemes rewarding more consensus nodes at the same time are also possible (e.g., [353]). While transaction fees are specified as part of a particular transaction, the block creation reward is usually part of the first transaction in the block, which is also referred to as *coinbase* transaction.

So far, it seems that no application utilizing or running on *public* blockchains has been able to work without introducing crypto-tokens (i.e., an incentive scheme), even if the use case is not financial in nature, e.g., e-voting [249], notaries [55], [178], secure timestamping [344], [157], reputation systems [85], social networks [97]. The situation is different in the context of private (permissioned) blockchains, which are usually provisioned by a single organization or a consortium and do not need to introduce their own crypto-tokens to operate. Misaligned incentives can cause consensus-level vulnerabilities, e.g., when it becomes profitable to cause the blocks of other nodes to be dropped in order to claim higher mining rewards [142] or transaction fees [87]. The design of incentive mechanisms is a research field by itself [228], and will not be treated further in this paper unless there is a clear relationship with a security vulnerability.

E. Basis of Consensus Protocols

In general, two marginal techniques deal with the establishment of a consensus among consensus nodes: *lottery* and *voting* [192]. In addition to them, combinations of these techniques, which leverage the advantages of both groups, have become popular.

Lottery-Based Protocols. Approaches from the lottery group aim to provide consensus by running a lottery that elects a leader or a committee, which then decides on the transaction order. The advantages of lottery-based approaches are: a small network traffic overhead and high scalability, since the process is usually non-interactive (e.g., Nakamoto consensus in Bitcoin [265], Proof-of-Elapsed-Time [95], Ouroboros [211]). However, a disadvantage of this approach is the possibility that multiple winners can be elected and propose conflicting blocks with the same height, which naturally leads to inconsistencies called *forks*. For this reason, fork-choice rules must be specified to resolve such forks. There are two types of fork-choice rules that are also referred to as branch difficulty computation. For the *longest chain rule*, the chain with the largest number of blocks is selected in the case of a conflict, while for the case of *strongest chain rule*, the selection also involves a quality of each block and hence their chain (e.g., [341], [342], [355], [390], [353]). Note that the possibility of forks and the need for their resolution in this category of protocols causes an increase of the time to finality, which is a negative aspect of these protocols.

Voting-Based Protocols. In this group of protocols, the agreement on transactions is reached through the votes of all participants; hence, a block may be produced only collaboratively. Instances of this category include Byzantine fault tolerant (BFT) protocols – which require the consensus of a majority quorum (usually, 2/3) of all consensus nodes (e.g., [89], [18], [69], [329], [210], [124], [247], [79]). The advantage of approaches from this category is low-latency finality (i.e., the elapsed time after a block becomes irreversible) and high processing throughput. Note that low-latency finality is achieved due to a negligible likelihood of forks in this group of protocols. On the other hand, the protocols from this group suffer from low scalability, thus, their throughput forms a trade-off with scalability (i.e., the higher the number of nodes, the lower the throughput).

Combinations. A combination of the previous two groups is often used to benefit from their pros. To reach scalability, it is desirable to shrink the number of active consensus nodes that participate in the lottery approach, so that only nodes of this smaller group (i.e., committee) vote for a final block (e.g., [155], [114], [172], [403], [210]). Another option to reduce the total number of voting nodes is to split them into several groups (a.k.a., shards) that run a consensus protocol in parallel (e.g., [217], [388]). Such a setting further increases the throughput in contrast to the single-group option, but on the other hand, requires a mechanism that accomplishes inter-shard transactions.

F. Failure Models in Distributed Consensus Protocols

The relevant literature mentions two main failure models for consensus protocols [326], while only the second one is more important for blockchains:

Fail-Stop Failures: A node either stops its operation or continues to operate, while obviously exposing its faulty behavior to other nodes. Hence, all other nodes are aware of the faulty state of that node (e.g., tolerated in Paxos [225], Raft [283], Viewstamped Replication [282]).

Byzantine Failures: In this model, the failed nodes (a.k.a., Byzantine nodes) may perform arbitrary actions, including malicious behavior targeting the consensus protocol and collusions with other Byzantine nodes. Hence, the Byzantine failure model is of particular interest to security-critical applications, such as blockchains (e.g., Nakamoto's consensus [265], pure BFT protocols [89], [18], [69], [79], and hybrid protocols [155], [217], [388]).

IV. The Security Reference Architecture

We present two models of the security reference architecture, which facilitate systematic studying of vulnerabilities and threats related to the blockchains and application running on top of them. First, we introduce the stacked model, which we then project into the threat-risk assessment model.

A. Stacked Model

To classify security aspects of blockchains, we utilize a stacked model consisting of four layers (see Figure 3). A similar stacked model was already proposed in the literature [373]. In contrast to [373], we preserve only such a granularity level that enables us to isolate security threats and their nature. Moreover, we propose a novel categorization of blockchain applications as part of the application layer. In the following, we briefly describe each layer.

(1) The network layer (see Section V) consists of the data representation and network services planes. The data representation plane deals with the storage, encoding, and protection of data, while the network service plane contains the discovery and communication with protocol peers, addressing, routing, and naming services.

(2) The consensus layer (see Section VI) deals with the ordering of transactions, and we divide it into three main categories according to the protocol type: Byzantine Fault Tolerant (e.g., [89], [38], [77], [124], [256]), Proof-of-Resource (e.g., [265], [125], [287], [253], [303]), and Proof-of-Stake (e.g., [33], [211]) protocols.

(3) The replicated state machine (RSM) layer (see Section VII) deals with the interpretation of transactions, according to which the state of the blockchain is updated. In this layer, transactions are split into parts, where the first part deals with the privacy of data in transactions as well as the privacy of the users who created them, and the second part – smart contracts – deals with security and safety aspects of decentralized code execution in this environment.

(4) The application layer contains the most common end-user functionalities and services. We divide this layer into two groups. The first group (see Section VIII) represents the applications that provide common functionalities for most of the higher-level blockchain applications, and it contains the following categories: wallets, exchanges, oracles, filesystems, identity management, and secure timestamping. We refer to this group as applications of the blockchain ecosystem. The next group (see Section IX) of application types resides on the higher level and focuses on providing certain end-user functionality. This group contains categories such as e-voting, notaries, identity management, auctions, escrows, etc. We found out that these higher level applications inherit security aspects from particular categories in the underlying ecosystem group (see Figure 14).

Throughout the paper, we summarize components and categories of particular layers of the reference architecture with their respective security threats, their origin, and corresponding countermeasures and/or mitigation techniques. Moreover, we discuss the pros and cons of particular categories within the first three layers of our stacked model – see the brief summary in Table I.[3]

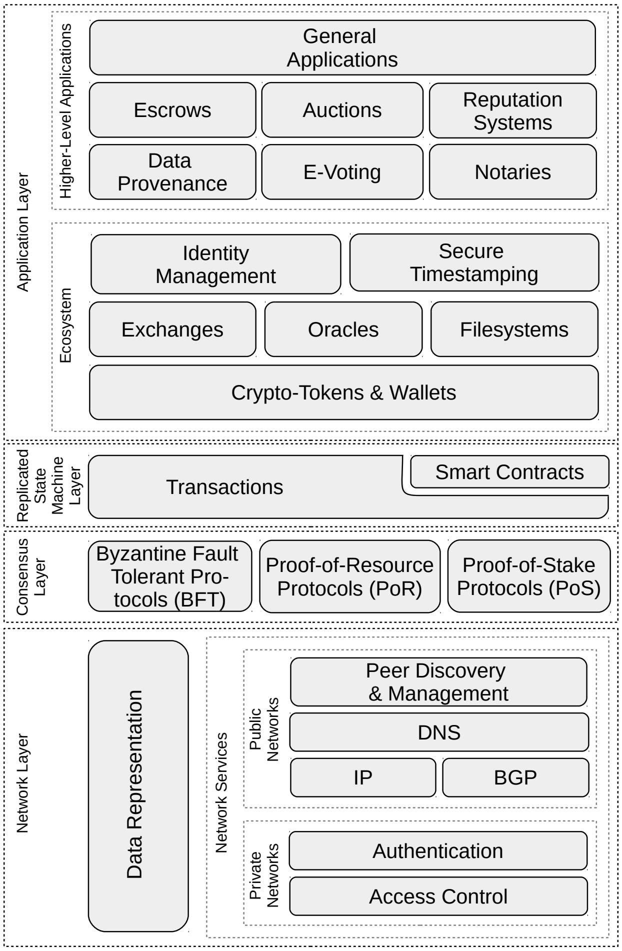

Figure 3: Stacked model of the security reference architecture.

B. Threat-Risk Assessment Model

To better capture the security-related aspects of blockchain systems, we introduce a threat-risk model (see Figure 4) based on the template of ISO/IEC 15408 [105]. This model includes the following components and actors:

Owners are blockchain users who run any node type. Owners posses crypto-tokens, and they might use or provide blockchain-based applications and services. Additionally, they involve consensus nodes that earn crypto-tokens from running the consensus protocol.

Assets consist of monetary value (i.e., crypto-tokens or other tokens), blockchain functionalities, as well as application-layer services built on top of them (e.g., notaries, escrows, data provenance, auctions). The authenticity of users, availability of services, the privacy of users, and the privacy of data might also be considered as application-specific assets. Furthermore, we include here availability of services and applications built on top of blockchains as well as the reputation of their providers.

[3] Due to its extent and complexity, we do not provide an explicit summary of the pros and cons of the application layer, but instead, we loosely mention them across descriptions of particular categories in this layer.

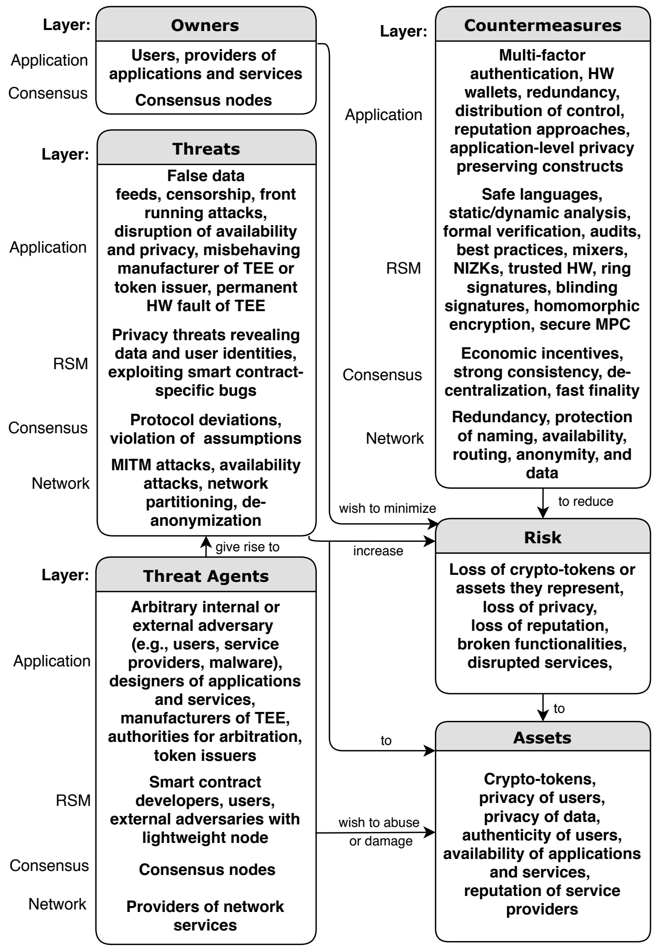
Layer:
Owners
Application
Users, providers of applications and services
Consensus
Consensus nodes
Layer:
Threats
Application
False data feeds, censorship, front running attacks, disruption of availability and privacy, misbehaving manufacturer of TEE or token issuer, permanent HW fault of TEE
RSM
Privacy threats revealing data and user identities, exploiting smart contract-specific bugs
Consensus
Protocol deviations, violation of assumptions
Network
MITM attacks, availability attacks, network partitioning, de-anonymization
give rise to
Layer:
Threat Agents
Application
Arbitrary internal or external adversary (e.g., users, service providers, malware), designers of applications and services, manufacturers of TEE, authorities for arbitration, token issuers
RSM
Smart contract developers, users, external adversaries with lightweight node
Consensus
Consensus nodes
Network
Providers of network services
Layer:
Countermeasures
Application
Multi-factor authentication, HW wallets, redundancy, distribution of control, reputation approaches, application-level privacy preserving constructs
RSM
Safe languages, static/dynamic analysis, formal verification, audits, best practices, mixers, NIZKs, trusted HW, ring signatures, blinding signatures, homomorphic encryption, secure MPC
Consensus
Economic incentives, strong consistency, de-centralization, fast finality
Network
Redundancy, protection of naming, availability, routing, anonymity, and data
to reduce
wish to minimize
increase
Risk
Loss of crypto-tokens or assets they represent, loss of privacy, loss of reputation, broken functionalities, disrupted services,
to
to
Assets
wish to abuse or damage
Crypto-tokens, privacy of users, privacy of data, authenticity of users, availability of applications and services, reputation of service providers

Figure 4: Threat-risk assessment model of the security reference architecture.

Threat agents are mostly malicious users whose intention is to steal assets, break functionalities, or disrupt services. However, threat agents might also be inadvertent entities, such as developers of smart contracts who unintentionally create bugs and designers of blockchain applications who make mistakes in the design or ignore some issues.

Threats facilitate various attacks on assets. Threats arise from vulnerabilities at the network, in smart contracts, from consensus protocol deviations or violations of consensus protocol assumptions, or application-specific vulnerabilities.

Countermeasures protect owners from threats. They involve various security and safety solutions and tools, incentives, reputation techniques, best practices, etc.

Risks are caused by threats and their agents and may lead to a loss of monetary assets, a loss of privacy, a loss of reputation, service malfunctions, and disruptions of services and applications (i.e., availability issues).

The owners wish to minimize the risk caused by threats that arise from threat agents. Within our stacked model, different threat agents appear at each layer. At the **network layer**, there are service providers including parties managing IP addresses and DNS names. The threats at this layer arise from man-in-the-middle (MITM) attacks, network partitioning, de-anonymization, and availability attacks. Countermeasures contain protection of availability, naming, routing, anonymity, and data. At the **consensus layer**, consensus nodes may be malicious and wish to alter the outcome of the consensus protocol by deviating from it. Moreover, if they are powerful enough, malicious nodes might violate assumptions of consensus protocols to take over the execution of the protocol or cause its disruption. The countermeasures include well-designed economic incentives, strong consistency, decentralization, and fast finality solutions. At the **RSM layer**, the threat agents may stand for developers who (un)intentionally introduce semantic bugs in smart contracts (intentional bugs represent backdoors) as well as users and external adversaries running lightweight nodes who pose threats due to the exploitation of such bugs. Countermeasures include safe languages, static/dynamic analysis, formal verification, audits, best practices, and design patterns. Other threats of the RSM layer are related to compromising privacy of data and user identities with mitigation techniques involving mixers, privacy-preserving cryptography constructs (e.g., non-interactive zero-knowledge proofs (NIZKs), ring signatures, blinding signatures, homomorphic encryption) as well as usage of trusted hardware (respecting its assumptions and attacker models declared). At the **application layer**, threat agents are broad and involve arbitrary internal or external adversaries such as users, service providers, malware, designers of applications and services, manufactures of trusted execution environments (TEE) for concerned applications (e.g., oracles, auctions), authorities in the case of applications that require them for arbitration (e.g., escrows, auctions) or filtering of users (e.g., e-voting, auctions), token issuers. The threats on this layer might arise from false data feeds, censorship by application-specific authorities (e.g., auctions, e-voting), front running attacks, disruption of the availability of centralized components, compromising application-level privacy, misbehaving of the token issuer, misbehaving of manufacturer of TEE or permanent hardware (HW) faults in TEE. Examples of mitigation techniques are multi-factor authentication, HW wallets with displays for signing transactions, redundancy/distributions of some centralized components, reputation systems, and privacy preserving-constructs as part of the applications themselves. We elaborate closer on vulnerabilities, threats, and countermeasures (or mitigation techniques) related to each layer of the stacked model in the following sections.

Involved Parties & Blockchain's Life-Cycle. In the background section (see Section III) we presented several types of involved parties in the blockchain infrastructure (see Figure 1). We emphasize that these parties are involved in the operational stage of the blockchain's life-cycle. However, in the design and development stages of the blockchain's life-cycle, programmers and designers should also be considered as potential threat agents who influence the security aspects of the whole blockchain infrastructure (regardless of whether their intention is malicious or not). This is of great concern especially for applications built on top of blockchains (i.e., at the application layer) since these applications are usually not reviewed by the community or public, as it is typical for other (lower) layers.

Layer			Category in a Layer	Pros	Cons
Network Layer			**Private Networks**	● low latency, high throughput ● centralized administration, ease of access control ● privacy of data, privacy of identities ● meeting regulatory obligations ● resilience to external attacks	● VPN is required for geographically spread participants ● suitable only for permissioned blockchains ● insider threat at nodes with administrative privileges
			Public Networks	● high decentralization ● high availability ● openness & low entry barrier (low cost of broadband connection, resistance to regulations)	● high and non-uniform latency ● single point-of-failure (DNS, IP, and ASes are managed by centralized parties) ● external adversaries (botnets, compromised BGP/DNS servers) ● stolen identities
Consensus Layer			**PoR**	● high cost of overriding the history of blockchain ● high scalability	● high operational costs ● low throughput ● low finality
			BFT	● high throughput ● fast finality	● low scalability ● high communication complexity ● limited number of nodes (efficient use only in permissioned blockchains)
			PoS	● energy efficiency	● PoS specific attacks and issues ● supports only semi-permissionless setting ● slow finality
			PoS+BFT	● energy efficiency ● higher scalability ● probabilistic security guarantees ● lower communication overhead than BFT	● some PoS specific attacks ● supports only semi-permissionless setting
			PoR+BFT	● higher scalability than BFT ● fast finality	● spending some scarce resources
			PoR+PoS (i.e., PoA)	● high scalability	● spending some scarce resources ● some PoS specific attacks ● slow finality
Replicated State Machine Layer	Transactions	Protecting Identities	**Standard Approach**	● fast processing ● ease of verification	● identities are only pseudonymous and can be traced to IPs ● all data of transactions are publicly visible
			Standard Approach + Mixers	● privacy identity protection of users in a group ● ease of verification	● additional complexity, in some cases unlinkability by the mixer or involved parties in a group ● all data of transactions are publicly visible
			NIZKs and Ring-Signatures	● identities are annonymized to the extend of the group	● additional computation overhead for running the schemes
			Multiparty Computation Blindig Signatures, Layered-Encryption	● unlinkability for all involved parties	● additional computation overhead for running the schemes
		Protecting Data	**NIZKs, Blinding Signatures, Homomorphic Encryption**	● privacy of data in cryptocurrency platforms	● additional computation overhead for running the schemes
			Trusted Transaction Managers, Trusted Hardware, Multiparty Computations	● privacy of data in transactions of smart contract platforms	● additional computation overhead for running the schemes
	Smart Contracts		**Turing-Complete Languages**	● smart contracts may contain an arbitrary programming logic	● wide surface for making the programming bugs that often results in vulnerabilities
			Turing-Incomplete Languages	● small attack surface and emphasis on safety	● the programming logic serves only for limited purposes

Table I: Pros and cons of various categories within the first three layers of our proposed stacked model.

V. Network Layer

To communicate, blockchains usually introduce *peer-to-peer overlay networks* built on top of other networks (see Figure 5); hence, blockchains inherit security and privacy issues from their underlying networks. Based on permission to join the blockchain system, the networks are either private or public. A private network is a network of local devices whose access is insulated from public networks. The Internet is a public network of interconnected autonomous systems (ASes) that relay network traffic at their borders. The network layer is divided into data representation and network sub-planes (see Figure 3). The data representation plane is protected by cryptographic primitives that ensure data integrity, user authentication, and optionally confidentiality, privacy, anonymity,

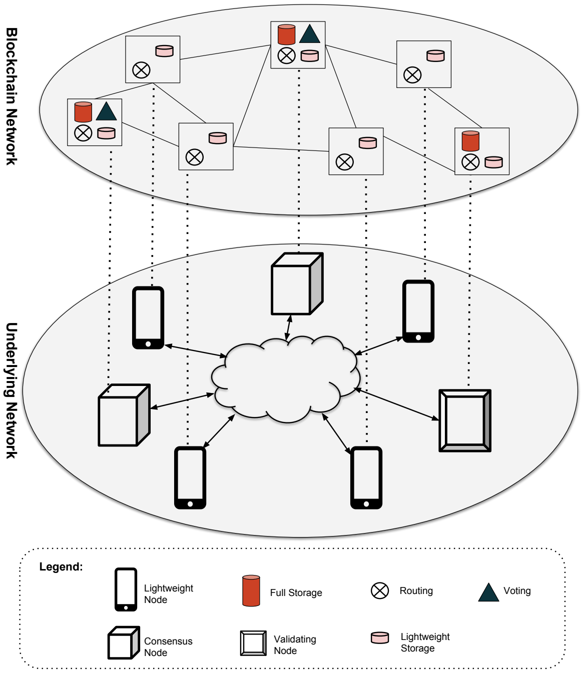

Figure 5: Blockchains are overlay networks over a private or a public network infrastructures.

non-repudiation, and accountability. The main blockchain-oriented services provided by the network layer are peer management and discovery, which rely on the internals of the underlying network, such as domain name resolution (i.e., DNS) or network routing protocols (e.g., LAN routing for IP, WAN routing such as BGP). In the following, we discuss pros and cons of private and public networks and their security threats that affect blockchains.

A. *Private Networks*

A private network ensures low latency, a centralized administration, privacy, and meeting regulatory obligations (e.g., HIPAA[4] for healthcare data). The organization owning the network provides access to local participants as well as to external ones when required; hence systems deploying private networks belong to the group of permissioned private blockchains. The inherent feature of private networks is that authentication and access control can be provided at the network layer.

1) Pros: Access control is achieved by centralized authentication of users and assigning them roles. A private network has full control over routing paths and physical resources used, which enables suitable regulation of the network topology and transmission medium for the given requirements. *Data privacy* is ensured by permissioned settings. *User identities* are only revealed within a private group of nodes. They are immune to external attacks in contrast to public networks (see Section V-B3). *Fine-grained authorization controls* are applied by the operators of the network resources to implement the security principle of minimal exposure and thus mitigate insider threat attacks on a local network. *Resource availability* is easier to manage and foresee, as all network participants and the deployment scenario are known ahead of time.

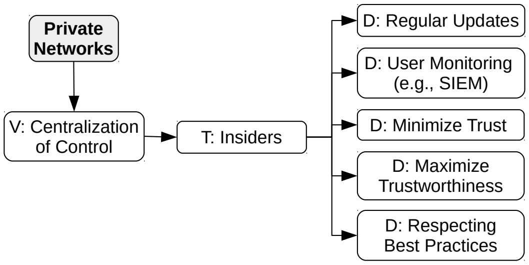
Private Networks
V: Centralization of Control
T: Insiders
D: Regular Updates
D: User Monitoring (e.g., SIEM)
D: Minimize Trust
D: Maximize Trustworthiness
D: Respecting Best Practices

Figure 6: Vulnerabilities, threats, and defenses in private networks (network layer).

2) Cons: Virtual Private Network (VPN) connectivity is required to communicate between private networks spread over different geographical locations. While VPNs are in general secure, they inherit the disadvantages of running a service over the Internet. *Applicability* of private networks is suitable only for permissioned and private blockchains.

3) Security Threats and Countermeasures: We present a taxonomy of threats on the private networks of the blockchains, their origins, and defenses against them in Figure 6. In the following, we describe these attacks as well as possible defense techniques. *Insiders* may pose a serious threat to security [16]. A compromised node may already have administrative privileges or it may obtain them by exploiting system, network, or security vulnerabilities. *Countermeasures* include regular software updates, user monitoring (e.g., SIEM [349]), prevention techniques that minimize trust and maximize trustworthiness, as well as respecting best practices [104].

B. *Public Networks / the Internet*

Public networks provide high decentralization, openness, and low entry barrier, while network latency, privacy, and network control are put aside. These networks are naturally required by all public (permissionless) blockchain systems.

1) Pros: High availability is attractive to multi-homed nodes since they have alternate routes to send and receive messages. Multi-homed nodes may benefit from disseminating blocks across multiple channels, thereby increasing the chance of blocks being appended to the blockchain. *High decentralization* is achieved through geographical dispersion of nodes. Public peer-to-peer (p2p) networks are harder to shut down [316]. *Openness and low entry barriers* on the Internet are achieved through wide adoption, technology interoperability (e.g., using TCP/IP), economic (e.g., low cost of broadband connection) and societal (e.g., resistance to regulations) factors [62]. Statistical resource sharing [250] and openness are fundamental to a low entry barrier.

2) Cons: Single-point-of-failure – DNS with its hierarchy, IP addresses, and ASes are managed by centralized parties – Internet Corporation for Assigned Names and Numbers (ICANN); in particular, Internet Assigned Numbers Authority (IANA). *External adversaries* pose a threat to public networks. These adversaries can be classified based on their capabilities to which the blockchain network may be exposed [318]: (1) resources under attacker control (e.g., botnets, DNS and BGP servers), (2) identities are stolen or masqueraded (e.g., IP

[4] Health insurance portability and accountability act https://hipaa.com/.

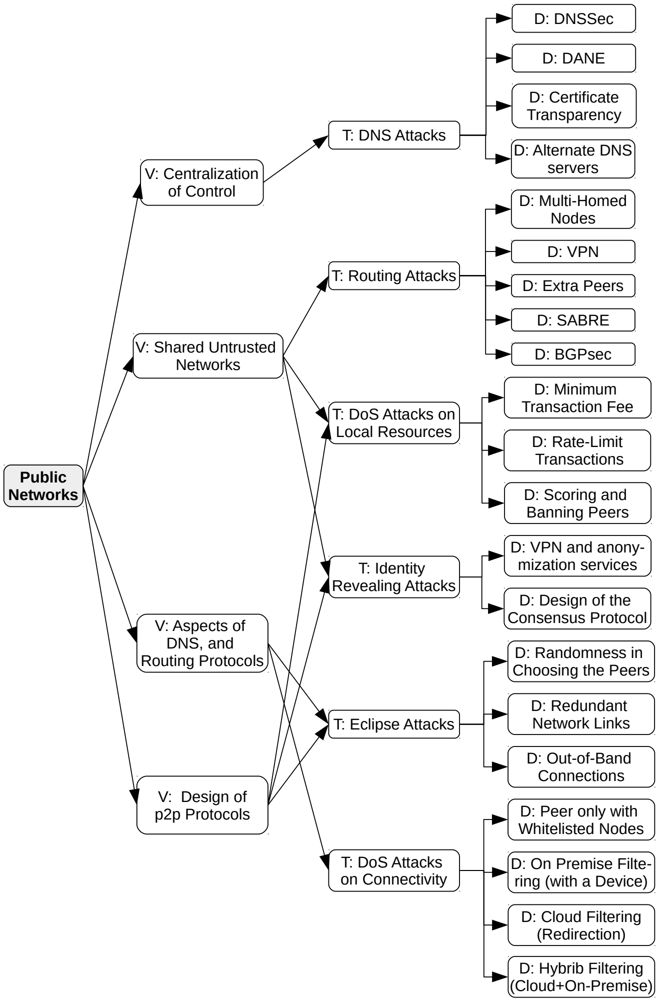
Public Networks
V: Centralization of Control
V: Shared Untrusted Networks
V: Aspects of DNS, and Routing Protocols
V: Design of p2p Protocols
T: DNS Attacks
T: Routing Attacks
T: DoS Attacks on Local Resources
T: Identity Revealing Attacks
T: Eclipse Attacks
T: DoS Attacks on Connectivity
D: DNSSec
D: DANE
D: Certificate Transparency
D: Alternate DNS servers
D: Multi-Homed Nodes
D: VPN
D: Extra Peers
D: SABRE
D: BGPsec
D: Minimum Transaction Fee
D: Rate-Limit Transactions
D: Scoring and Banning Peers
D: VPN and anonymization services
D: Design of the Consensus Protocol
D: Randomness in Choosing the Peers
D: Redundant Network Links
D: Out-of-Band Connections
D: Peer only with Whitelisted Nodes
D: On Premise Filtering (with a Device)
D: Cloud Filtering (Redirection)
D: Hybrib Filtering (Cloud+On-Premise)

Figure 7: Vulnerabilities, threats, and defenses in public networks (network layer).

addresses participating in an eclipse attack or route manipulation), (3) MITM attacker (i.e., eavesdropping and spoofing), (4) common vulnerabilities leading to exploits (e.g., observed in DNS BIND software [163]), (5) revealing secrets (e.g., deanonymizing peers). *Efficiency* – although an average Internet bandwidth has improved in recent years [4], a distribution of powerful infrastructure is not uniform, which results in a different latency among peers, and the overall latency of the network is increased – this, in turn, may result in the loss of created blocks and thus wasting consensus power.

3) Security Threats and Countermeasures: We present a taxonomy of the attacks on the public networks of the blockchains, their origins, and defenses against them in Figure 7, while in Table II, we list several evidences of incidents occurred in practice. In the following, we describe these attacks as well as possible defense techniques.

DNS attacks commonly arise from cache poisoning [343] that mainly affects nodes employing DNS bootstrapping [49] to retrieve online peers. One *countermeasure* is a security extension of DNS, called DNSSEC, which provides authentication and data integrity. In addition to standard DNS, name resolution can also be made using alternate DNS servers [110], [64].

Routing attacks are traffic route diversions, hijacking, or DoS attacks. Beside simple data eavesdropping or modification, these attacks may lead to network partitioning, which in turn raises the risks of 51% attacks or selfish mining attacks (see Section VI). *Countermeasures* are multi-homed nodes (or using VPN) for route diversity, choosing extra peers whose connections do not pass through the same ASes, preference of peers hosted on the same AS within the same /24 prefix (to reduce risk of partitions), and fetching the same block from multiple peers [14]. Another mitigation is SABRE [13], a secure relay network that runs alongside with the Bitcoin network. The BGPsec [229] is a security extension for BGP used between neighboring ASes, and it provides assurance of route origin and propagation by cryptographic verification.

Eclipse attacks aim to hijack all connections of a node to the blockchain network. Consequently, all traffic received and sent by the node is under the full control of the attacker. Eclipse attacks arise from threats on DNS and routing in the network, and they may be a result of vulnerabilities in p2p protocols [177], [381], [244]. Eclipse attacks increase chances of selfish mining and double spending attacks (see Section VI) – the eclipsed victims may vote for an attacker's chain. *Countermeasures:* Improving randomness in choosing peers was proposed in paper [177] by several rules that manage the peer table. Another mitigation strategy against eclipse attacks is to use redundant network links or out-of-band connections to verify transactions (e.g., by a blockchain explorer). Also, note that countermeasures for DNS and routing attacks are applicable here as well.

DoS attacks on connectivity of consensus nodes may result in a loss of consensus power, thus preventing consensus nodes from being rewarded [184]. For validating nodes, this attack leads to a disruption of some blockchain dependent services [334]. *Countermeasures:* One mitigation is to peer only with white-listed nodes. Methods to prevent volumetric DDoS include on-premise filtering (i.e., with an extra network device), cloud filtering (i.e., redirection of traffic through a cloud when DDoS is detected or through a cloud DDoS mitigation service), or hybrid filtering [15] (i.e., combinations of the previous two).

DoS attacks on resources such as memory and storage, may reduce the peering and consensus capabilities [230] of nodes. An example attack is flooding the network with low fee transactions (a.k.a., penny-flooding), which may cause memory pool depletion, resulting in a system crash. A possible mitigation is raising the minimum transaction fee and the rate-limit to the number of transactions. Several mitigating techniques are applied to Bitcoin [51] nodes including scoring DoS attacks and banning misbehaving peers. DoS attacks may also target (additional) centralized elements of blockchain infrastructure, such as servers communicating with hosted wallets (see Section VIII-A), which in turn might lead to application layer attacks targeting clients of the wallets.

Acronym / Incident	Threat / Vulnerability	Reference	Description	Impact ($)
Canadian Bitcoin hijack (February 2013)	BGP prefix hijacking (routing attack & eclipse attack)	[166]	Intercepted data between Bitcoin miners and Bitcoin mining pools. Tricked honest miners mined on an attacker controlled pool.	83,000
Bitcoin deanonymization (March 2015)	Sybil attack (identity revealing attacks)	[81]	A large number of fake nodes were introduced to deanonymize client traffic.	—
MyEtherWallet hijack (April 2018)	BGP hijacking (routing attack)	[146], [297]	A BGP hijack have been performed against Amazon DNS, leading to the wallet's server domain name being resolved to a Russian phishing site in several cities.	152,000
Electrum DoS (August 2019)	Volumetric DoS	[82]	DoS of legitimate Electrum servers was conducted as an attempt to get connected vulnerable Electrum wallets to a malicious server, which resulted in a loss of wallet funds.	In millions
Bitcoin partitioning (proof-of-concept)	Prefix hijacking (routing attack)	[12]	An attack to parition Bitcoin network by hijacking IP prefixes.	—
Erebus (proof-of-concept)	Malicious ISP attack (routing attack)	[363]	ISP uses its toplogical advantage to launch a stealthy partition attack.	—
Eclipse attack on Bitcoin (proof-of-concept)	Un-authenticated and unreliable peers	[177]	A view of the network by eclipsed peers is fully under the attacker's control.	—
De-anonymization in Bitcoin with Tor (proof-of-concept)	Abusing Bitcoin DoS protection (identity revealing attack)	[42]	Bitcoin peers were forced to ban Tor exit nodes of attacker's choice by abusing Bitcoin DoS protection. Thus, the attacker was able to control all remaining Tor exit nodes and cause client traffic to pass through them.	—
Suspected Penny flooding on Bitcoin (December 2017)	DoS on a mempool (attacking local resources)	[386]	Flooding a mempool with low fee transactions resulted into clogged up memory of consensus nodes and increased transaction processing fees.	—
2X-Mempool-Attack on Bitcoin (suspected)	Flooding by high-fee transactions (DoS on processing of legitimate transactions)	[387]	Mempool is flooded by high-fee transactions, preventing from regular-fee transactions being processed timely.	—

Table II: Incidents occurred at the network layer (public networks).

Identity revealing attacks are conducted by linking the IP address of a node with an identity propagated in transactions [42], [255]. Traffic analysis using Sybil listeners can reveal the linkage of node IP addresses and their transactions [81]. *Countermeasures* include using VPNs or anonymization services, such as Tor. See Section VII-A1 for further identity and privacy-protecting mechanisms at the RSM layer.

VI. Consensus Layer

The consensus layer of the stacked model deals with the ordering of transactions, while the interpretation of them is left for the RSM layer (see Section VII). The consensus layer includes three main categories of consensus protocols with regard to different principles of operation and thus their security aspects. First, we focus on the security aspects that are generic to all categories of consensus protocols, and then we detail each category.

A. Generic Attacks

We present a taxonomy of the generic attacks to all types of consensus protocols, their origins, and defenses against them in Figure 8. In the following, we describe these attacks as well as possible defense techniques.

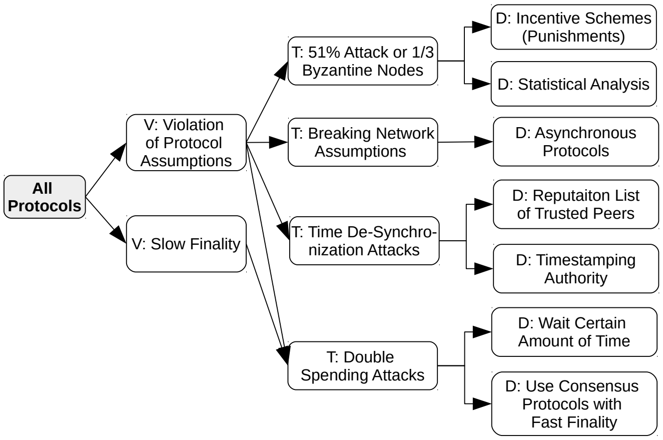
All Protocols
V: Violation of Protocol Assumptions
V: Slow Finality
T: 51% Attack or 1/3 Byzantine Nodes
T: Breaking Network Assumptions
T: Time De-Synchronization Attacks
T: Double Spending Attacks
D: Incentive Schemes (Punishments)
D: Statistical Analysis
D: Asynchronous Protocols
D: Reputaiton List of Trusted Peers
D: Timestamping Authority
D: Wait Certain Amount of Time
D: Use Consensus Protocols with Fast Finality

Figure 8: Generic threats and defenses of the consensus layer.

1) Violations of Protocol Assumptions:

Adversarial Centralization of Consensus Power. In these attacks, a design assumption about the decentralized distribution of consensus power is violated. Examples of this category are *51% attacks* for PoR and PoS protocols as well as $\frac{1}{3}$ of *Byzantine nodes* for BFT protocols (and their combinations). In a 51% attack, the majority of the consensus power is held by the adversary, thus also the result of the protocol is under its control. In *Byzantine attacks*, a quorum of $\frac{1}{3}$ adversarial consensus nodes might cause the protocol being disrupted or even halted. As a design-oriented countermeasure, it is important to promote decentralization by incentive schemes that reward honest participation and discourage [254] or punish [75], [114] protocol violations.

Breaking Network Assumptions: Protocols assuming synchronous or partially synchronous network delivery would inevitably fail when this assumption does not hold. For instance, this assumption can be violated in BFT protocols by *an unpredictable network scheduler*, as demonstrated on PBFT protocol [256]. This fact motivates asynchronous BFT protocols that can be based on threshold-based cryptography, which enables reliable and consistent broadcast [79] [256].

Time De-Synchronization Attacks. Usually, besides system time, nodes in PoW and PoS maintain network time that is computed as the median value of the time obtained from the peers. Such a time is often put into the block header, while nodes, upon receiving a block, validate whether it fits freshness constraints. An attacker can exploit this approach by connecting a significant number of nodes and propagate inaccurate timestamps, which can slow down or speed up the victim node's network time [61]. When such a desynchronized node creates a block, this block can be discarded by a network due to freshness constraints. To avoid de-synchronization attacks, a node can build a reputation list of trusted peers or employ a timestamping authority [352].

Double-Spending Attack. This attack is possible due to the creation of two or more conflicting blocks with the same height, resulting in inconsistencies called *forks*. Thus, some crypto-tokens might be temporarily spent in both conflicting blocks, while only a single block is later included in the honest chain. A double-spending attack mainly affects consensus protocols with slow finality. This attack usually occurs as a consequence of 51% attack.[5] To prevent this attack, it is recommended to wait a certain amount of time (i.e., a few next blocks) until a block "is settled"[6] or utilize consensus protocols with fast finality.

B. Proof-of-Resource Protocols (PoR)

Protocols from this category require nodes to prove a spending of a scarce resource in a lottery-based fashion [192]. Scarce resources may stand for: *(1) Computation* that is represented by Proof-of-Work (PoW) protocols (e.g., Bitcoin, Ethereum). *(2) Storage* used in the setting of Proof-of-Space protocols [125] (e.g., Spacecoin [291], SpaceMint [190]). *(3) Crypto-tokens* spent for Proof-of-Burn protocols [206] (e.g., Slimcoin [287]). *(4) Combinations and modification* of the previous types, such as storage and computation, called Proof-of-Retrievability (e.g., Permacoin [253]) and storage over time, which is represented by Proof-of-Space protocols (e.g., Filecoin [303]). Another hybrid example of this category is a combination of PoR with elapsed time, such as in PeerCoin [320]. However, it is a philosophical question whether we consider elapsed time as a resource that is spent or as a stake that is invested – note that literature often categorizes PeerCoin as the first instance of a (hybrid) PoS protocol, hence we incline to the second option.

PoR protocols belong to the first generation of consensus protocols, and they are mostly based on Nakamoto Consensus [265] that utilize PoW, inheriting its pros (e.g., high scalability) and cons (e.g., low throughput). For the detailed analysis of several PoW designs, we refer the reader to [395].

1) Pros: In PoR protocols, malicious overriding of the history of blockchain (or its part) requires spending at least the same amount of resources as was spent for its creation. This is in contrast to principles of PoS protocols, where a big enough coalition may override the history with almost no cost.

2) Cons: stand mainly for a high operational cost. Moreover, these protocols provide only probabilistic finality, which enables attacks forking the last few blocks of the chain.

3) Security Threats and Mitigations: We present a taxonomy of the attacks related to PoR protocols, their origins, and defenses against them in Figure 9, while in Table III, we list several evidences of incidents occurred in practice. In the following, we describe these attacks as well as possible defense techniques.

Selfish Mining: In selfish mining [142],[7] an adversary attempts to privately build a secret chain and reveal it to the public only when an honest chain is "catching up" with the secret one. The longest chain rule causes honest miners to adopt the attacker's chain and invalidate the honest chain, thus wasting their consensus power. This attack is more efficient when consensus power of a selfish miner reaches some threshold (e.g., 30%). The selfish mining strategy was later generalized [323] and extended to other variants that increase the profit of the attacker [270]. *Countermeasures:* (1) For the case of the longest chain rule, the first introduced mitigation is uniform tie breaking [142], which tells consensus nodes to choose the chain to extend uniformly at random, regardless of which one they received first. However, this technique is less effective when assuming network delays [323]. (2) As the longest chain rule enables this attack, it is recommended to use other fork choice rules that also account for the quality of solutions and make the decision deterministic, as opposed to a uniform tie breaking. An example of such a rule is to select the block based on the smallest hash value. Another example is to include partial solutions [390], [292], [353] or full (orphaned) blocks [342], [394] for computation of block's quality. (3) Another option for a deterministic fork choice rule is using a pseudo-random function [216], which moreover provides unpredictability,[8] hence an attacker

[5] In the case of PoR protocols, this attack may also occur as a consequence of the selfish mining attack.

[6] Note that this is an application layer mitigation, since the amount of time to wait is determined based on the use case.

[7] Note that selfish mining is theoretically possible even in PoS protocols [67], but requiring them to have predictable randomness for the leader election, which is usually a design-oriented vulnerability (see Section III-D). However, in PoR, selfish mining is possible even with unpredictable randomness for the leader election.

[8] As opposed to uniform-tie breaking that provides only unpredictability, this solution brings determinism additionally to unpredictability.

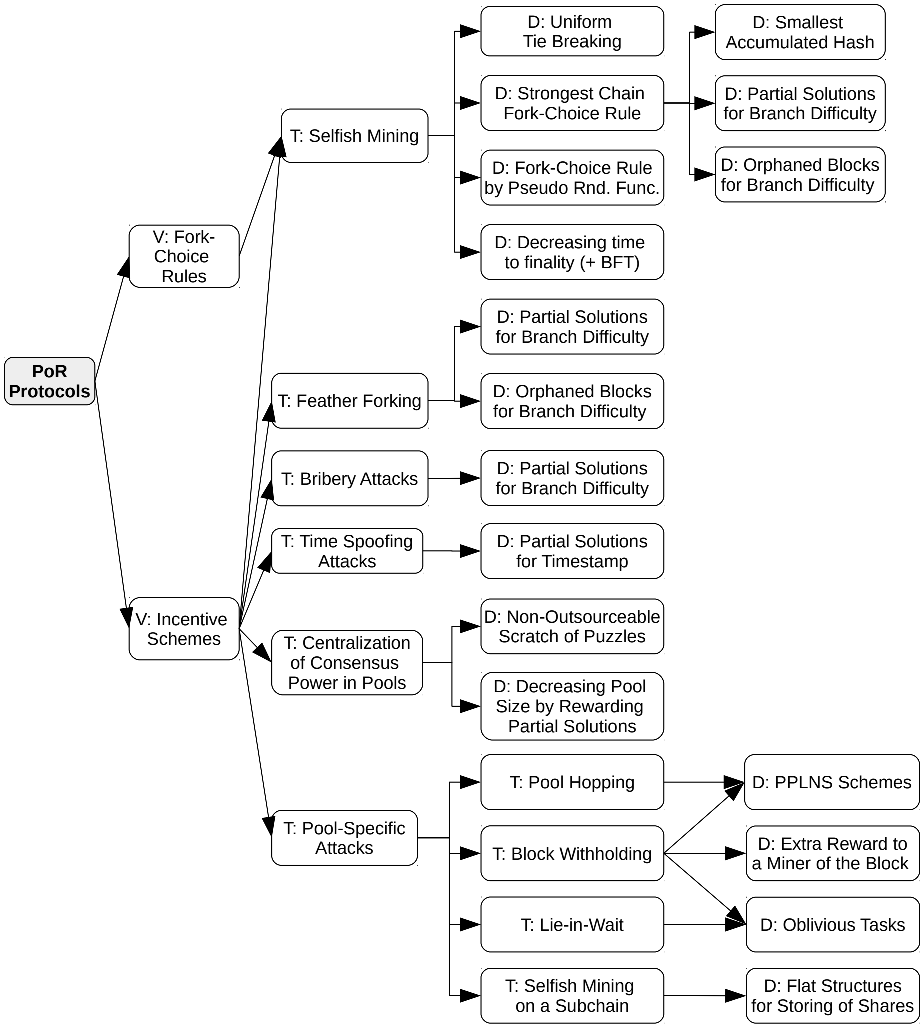
PoR Protocols
V: Fork-Choice Rules
V: Incentive Schemes
T: Selfish Mining
T: Feather Forking
T: Bribery Attacks
T: Time Spoofing Attacks
T: Centralization of Consensus Power in Pools
T: Pool-Specific Attacks
D: Uniform Tie Breaking
D: Strongest Chain Fork-Choice Rule
D: Fork-Choice Rule by Pseudo Rnd. Func.
D: Decreasing time to finality (+ BFT)
D: Smallest Accumulated Hash
D: Partial Solutions for Branch Difficulty
D: Orphaned Blocks for Branch Difficulty
D: Partial Solutions for Branch Difficulty
D: Orphaned Blocks for Branch Difficulty
D: Partial Solutions for Branch Difficulty
D: Partial Solutions for Timestamp
D: Non-Outsourceable Scratch of Puzzles
D: Decreasing Pool Size by Rewarding Partial Solutions
T: Pool Hopping
T: Block Withholding
T: Lie-in-Wait
T: Selfish Mining on a Subchain
D: PPLNS Schemes
D: Extra Reward to a Miner of the Block
D: Oblivious Tasks
D: Flat Structures for Storing of Shares

Figure 9: Vulnerabilities, threats, and defenses of PoR protocols (consensus layer).

cannot determine his chances to win a tie. (4) PoW protocols can be combined with BFT protocols, where PoW is used only for joining the protocol and BFT for consensus itself (e.g., [217], [388], [216], [403], [240]).

Feather Forking: In this attack [252], the adversary creates incentives for rational miners to collectively censor certain transactions. Before a mining round begins, an adversary announces that he will not extend the block containing blacklisted transactions, and thus will attempt to extend a forked chain. Although this strategy is not profitable for the adversary and the success rate is dependent on his consensus power, rational nodes prefer to join the censorship to avoid the potential loss. *Countermeasures:* design-oriented protection is to minimize the chance of the attacker being successful, which can be done by including (and rewarding) partial solutions [390], [315], [292], [353] or full orphaned blocks [342], [394] into branch difficulty computation.

Bribery Attacks: Whereas feather forking involves adversaries who try to influence the behavior of miners by threatening to hurt their profits, bribery attacks involve the offering of direct rewards to miners. For example, consensus nodes could be bribed to enable double-spending attacks [57] or to reorder transactions within a block, and thus enable the transaction front-running by other means than natural priority gas auctions (PGAs) [113].[9] *Countermeasures:* assuming that the miners who accept bribes constitute a minority, a possible mitigation technique is to utilize partial solutions [353], [390], [315], [292], which reduce a likelihood that the double-spending attacks succeed. Regarding "bribed" transaction front-running, the misbehavior happens entirely off-chain since miners have complete control over the transaction ordering process, so on-chain mitigation is challenging. Moreover, the likelihood of this attack is directly proportional to the consensus power of the bribed miner; hence, this attack is more feasible for mining pools. However, since no evidence confirming collusion between mining pools and bots has been found yet, mining pools are likely to be discouraged from accepting bribes due to the fear of consequences (e.g., a decrease in the market value of the

[9] In PGAs, users (arbitrage bots) compete with each other to be the first to interact with a smart contract (e.g., due to profit from intra-chain exchanges – see Section VIII-B).

crypto-tokens upon public disclosure of these attacks).

Time Spoofing Attacks: Time spoofing attacks target a time-based difficulty computation algorithm of PoR protocol with intention to decrease the difficulty of the puzzle, and thus minimize effort for obtaining the same reward. In particular, the attacker is a consensus node that mines blocks with delayed timestamps, which indicates that a puzzle is too hard to meet block creation rate and therefore difficulty needs to be decreased. *Countermeasures:* A solution that improves accuracy of the timestamps may utilize partial solutions found by all nodes into an averaged timestamp computation [353]. Note that impact of time spoofing attack might be significant also at the application layer of SRA, especially in use cases that rely in a timestamp accuracy (see Section VIII-F).

Pool Specific Attacks: Since PoR protocols are usually based on a lottery having a single winner [265], rewarding for participation imposes a high payout variance for solo miners (i.e., once in a few years). As a consequence, mining pools emerged and caused centralization of the mining power, which may result in selfish mining, double spending, or 51% attacks. *Countermeasures:* Non-outsourceable scratch-off puzzles [254] avoid creation of pools but require each consensus node to meet high demands on connectivity and storage, as opposed to centralized pools, where only a pool operator needs to meet these demands. If pools are acceptable, their size can be controlled by protocols that reward partial solutions [390], [315], [292], [353] and thus minimize payout variance. For a detailed analysis of rewarding schemes in pools, we refer the reader to [317]. In the following, we describe several types of pool specific attacks.

a) Pool Hopping. The individual contribution of miners in a pool is proved by broadcasting partial solutions, called *shares*. If pay-per-share (PPS) rewarding is employed (i.e., pool operator instantly rewards miners showing shares), an attacker may jump into another pool after his mining time in a victim pool reaches a certain threshold [310]; mining at the early stages of a round is statistically more profitable than mining at the end of the round. As a countermeasure pay-per-last-N-shares (PPLNS) scheme and its variants [328] can be used. PPLNS removes the concept of rounds and instead of immediate payments, it employs deterred payments after N submitted shares by a miner.

b) Block Withholding. An attacker may try to sabotage a victim pool – after mining a block in a victim pool, the attacker discards this block and continues mining at another pool [317]. Such withholding does not mean a direct gain for the attacker, but attacker may do a secret agreement with concurrent pool(s) that may reward the attacker for showing a withheld block [21] (a.k.a., sponsored block withholding). Mitigation for this kind of attack is (1) using of PPLNS scheme, (2) giving an extra reward to a miner of the block [22], (3) precluding miners from distinguishing between the share and a full solution (i.e., oblivious tasks). An example is represented by commitments specifying "the part of the expected solution," which are broadcast by the pool operator to the network at the beginning of each round [21].

c) Lie-in-Wait. If the miner finds a block in a victim pool, he does not immediately submit it to the pool operator, but instead focuses all his available mining power to the victim pool in order to increase his relative shares within a pool; after some time attacker releases the formerly found block. A countermeasure for this attack can be to use oblivious tasks – the miner is unable to distinguish between the full solution and share, e.g., [21].

d) Selfish Mining on a Subchain. Decentralized mining pools, such as p2pool [286], achieve decentralization by updating an intermediary coinbase transaction with mined shares. To keep a consistency with the previous versions of coinbase transaction within a mining round, its history is kept in a subchain. However, a chaining data structure enables selfish mining on a subchain, besides the fact that it implies a high stale rate of shares in a subchain.[10] A possible countermeasure is to use flat data structures for an aggregation of shares, such as a Merkle tree or hash of a set [353].

C. Byzantine Fault Tolerant (BFT) Voting Protocols

BFT protocols represent voting-based [192] consensus protocols that utilize Byzantine agreement and a state machine replication [326]. These protocols assume a fully connected topology, broadcasting messages, and a master-replicas hierarchy. Synchronous examples of this category are PBFT [89], RBFT [18], eventually synchronous examples are BFT-SMaRt [38], Tendermint [69], Byzantine Paxos [77], BChain [124], and asynchronous examples are SINTRA [79] and HoneyBadgerBFT [256]. For more details, we refer the reader to a review of BFT protocols and their practical applications in both permissioned and permissionless blockchains [80].

1) Pros: BFT protocols provide high throughput and a fast finality. To face their scalability limitation, BFT protocols are often combined with PoS or PoR. This is in line with a lottery approach [192] for selecting a portion of all nodes, referred to as committee, which further runs BFT consensus (e.g., Algorand [155], Zilliqa [403], DFINITY [172]).

2) Cons: The main con of traditional BFT protocols [77], [89] is a low scalability caused by a high communication complexity (i.e., $\Theta(n^2)$). Since these protocols can work efficiently only with a limited number of consensus nodes, they can be used in their pure form only in permissioned blockchains.

Issues with scalability and throughput can be resolved by applying cryptographic constructs [56], [78], [337] and partitioning consensus nodes into shards that process transactions in parallel [217], [388]. Another option is to prune the number of nodes running BFT into committees [155], which, however, reduces the security level of BFT and provides only

[10] Note that the same applies for Flux [390] and Subchains [315] that maintain a subchain but at the level of the whole network (as opposed to the level of the pool).

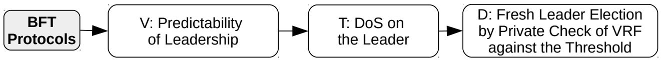
BFT Protocols
V: Predictability of Leadership
T: DoS on the Leader
D: Fresh Leader Election by Private Check of VRF against the Threshold

Figure 10: Vulnerabilities, threats, and defenses of BFT protocols (consensus layer).

probabilistic security guarantees depending on the committee size.

3) Security Threats and Mitigations: We present a taxonomy of the attacks related to BFT protocols, their origins, and defenses against them in Figure 10. In the following, we describe these attacks as well as possible defense techniques.

Denial of Service on a Leader: Since BFT protocols are mostly intended for (private) permissioned blockchains that are run by trusted participants, they do not assume an existence of malicious nodes whose goal is to sabotage the protocol. However, assuming such an adversary, a leader of the round might be DoS-ed since her leadership is known before the round starts, which causes a restart of the round. Moreover, this might be repeated until adversary's desired nodes are elected. *Countermeasures:* To prevent this attack, a node can privately determine whether it is a potential leader by using Verifiable Random Function (VRF) [155], and immediately release a block candidate; hence, after publishing this data, it is too late for a DoS attack on the node.

D. Proof-of-Stake Protocols (PoS)

Similar to the PoR category, PoS protocols are based on the lottery approach [192]. However, in contrast to PoR, no scarce resource is spent; instead, the nodes are required "to prove investment" of crypto-tokens in order to participate in a protocol, and thus potentially earn interest from the invested amount. The concept of PoS was for the first time proposed in the Bitcointalk forum [306]. The first technical realization of PoS is Peercoin [320], which is a combination with PoW – each node has its particular difficulty for PoW, which is based on the age of the coins a node owns. Although there exist a few pure PoS protocols (e.g., Chains of Activity [33], Ouroboros [211]), the trend is to combine them in a hybrid setting with PoR (e.g., Proof-of-Activity [35], Peercoin [320], Snow White [36]) or BFT protocols (e.g., Algorand [155]). In particular, a combination of PoS with BFT represents a promising approach, which takes advantages of both lottery and voting (i.e., scalability and throughput), while no resources are wasted.

1) Pros: The main feature of PoS protocols, as compared to PoR, is their energy efficiency. Although some PoS protocols are often combined with a PoR technique (e.g., [36], [320]), the overall energy spent is much less than in the case of pure PoR protocols.

2) Cons: Introduction of PoS protocols has brought PoS specific issues and attacks, while these protocols are still not formally proven to be secure. Next, PoS protocols are semi-permissionless – a node needs to first obtain a stake from any of the existing nodes to join the protocol.

3) Security Threats and Mitigations: We present a taxonomy of the attacks related to PoS protocols, their origins, and defenses in Figure 11. In the following, we describe these attacks as well as possible defense techniques.

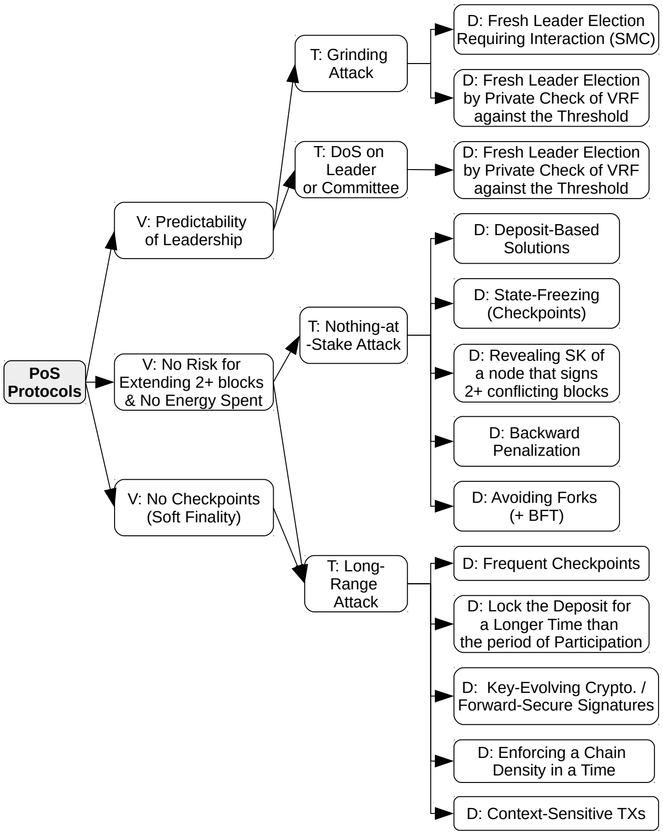
PoS Protocols
V: Predictability of Leadership
V: No Risk for Extending 2+ blocks & No Energy Spent
V: No Checkpoints (Soft Finality)
T: Grinding Attack
T: DoS on Leader or Committee
T: Nothing-at -Stake Attack
T: Long-Range Attack
D: Fresh Leader Election Requiring Interaction (SMC)
D: Fresh Leader Election by Private Check of VRF against the Threshold
D: Fresh Leader Election by Private Check of VRF against the Threshold
D: Deposit-Based Solutions
D: State-Freezing (Checkpoints)
D: Revealing SK of a node that signs 2+ conflicting blocks
D: Backward Penalization
D: Avoiding Forks (+ BFT)
D: Frequent Checkpoints
D: Lock the Deposit for a Longer Time than the period of Participation
D: Key-Evolving Crypto. / Forward-Secure Signatures
D: Enforcing a Chain Density in a Time
D: Context-Sensitive TXs

Figure 11: Vulnerabilities, threats, and defenses of PoS protocols (consensus layer).

Nothing-at-Stake: Since generating a block in PoS does not cost any energy, a node can extend two or more conflicting blocks without risking its stake, and hence increase its chance to be rewarded. Such behavior increases the number of forks and thus time to finality. *Countermeasures:* Deposit-based solutions (e.g., [75]) require nodes to make a deposit during some fixed period/round and checkpoint-based solutions (e.g., [75], [50], [114]) employ "state freezing" at periodic snapshots of the blockchain, while the blockchain can be reversed maximally up to the recent checkpoint. Another option is to punish a node that signs two conflicting blocks by embedding cryptographic solutions [231] enabling anybody to reveal identity and a private key of such a node. Another countermeasure is to use backward penalization of nodes that produced two or more conflicting chains [114], [75]. Finally, PoS protocols can be combined with BFT approaches, and thus forks can hardly occur (e.g., [155]).

Grinding Attack: If the leader or committee producing a block is determined before the round starts, then the attacker can bias this process to increase his chances of being selected in the future. For example, if a PoS protocol takes only a hash of the previous block for the election process, the leader of a block may bias a hash value by suitably adjusting the content of the block in a few attempts. *Countermeasures:* The grinding attack

Acronym / Incident	Threat / Vulnerability	Reference	Description	Impact ($)
Ethereum Classic (January 2019)	51% attack & double spending (violation of assumptions)	[76]	A 51% attack on Ethereum Classic led to a deep chain reorganization, which included double-spending attacks against Binance and Bitrue wallets.	1,100,000
Monacoin (May 2018)	51% attack & double spending (violation of assumptions)	[171]	A block reorganization on Monacoin included double-spending attacks that targeted several "Western exchanges" (particularly Livecoin). Unconfirmed reports on Reddit mention losses of 90,000 USD.	90,000
Bitcoin Gold (May 2018)	51% attack & double spending (violation of assumptions)	[257]	A Block withholding attack on Bitcoin Gold led to 76 double-spent transactions, mostly targeting cryptocurrency exchanges.	18,600,000
Litecoin Cash (May 2018)	51% attack & double spending (violation of assumptions)	[44]	A similar double-spending attack targeted Litecoin cash – the lead developer "Tanner" stated that he believes the mining power may have been rented.	—
Zencash (June 2018)	51% attack & double spending (violation of assumptions)	[391]	A 51% attack led to several block reorganizations, with the largest one that reversed 38 blocks. Only two transactions included double spending, but these were large enough to cause considerable losses.	550,000
Verge (April 2018)	51% attack & semantic bug (violation of assumptions)	[278]	A vulnerability in Verge's difficulty adjustment algorithm was used to amplify a 51% attack.	3,850,000
Verge (May 2018)	51% attack & semantic bug (violation of assumptions)	[330]	An inadequate response by Verge developers to the previous attack led to it being repeated a month later.	1,750,000
Checklocktime, CensorshipCon, GoldfingerCon, etc.	Bribery attacks	[200]	Various bribery attacks have been listed in the literature. In these attacks, it is profitable for consensus nodes to accept bribes for deviation from the protocol.	—

Table III: Incidents occurred at the consensus layer.

can be prevented by performing a fresh leader election by an interaction of nodes (e.g., the secure multiparty coin flipping protocol [211]) or by privately checking whether the VRF output is below a certain stake-specific threshold (e.g., [155]). The input of the VRF is the user's private key and the randomness unambiguously bound to the previous block; hence each consensus node computes the only VRF output during each round.

Denial of Service on a Leader/Committee: Alike in BFT protocols (see Section VI-C), if a leader or a committee is publicly determined before the round starts [211], then the adversary may conduct a DoS attack against them and thus cause a restart of the round – this might be repeated until adversary's desired nodes are elected. *Countermeasures:* A prevention technique was proposed in Algorand [155] – a node privately determines whether it is a potential leader (or committee member), and immediately releases a block candidate (or a vote) – hence, after publishing this data, it is too late for a DoS attack. The concept of VRF approach was also utilized in other protocols (e.g., [115], [172]).

Long-Range Attack: In this attack [73] (a.k.a., posterior corruption [114]), an adversary can "bribe" previously influential consensus nodes to sell their private keys or steal the private keys by other means. Since consensus nodes may exchange their crypto-tokens for fiat money anytime, selling their keys imposes no expenses and risk. If the attacker accumulates keys with enough stake in the past, he may rerun the consensus protocol and rewrite the history of the blockchain. A variant of long-range attack that considers transaction fee-based rewarding and infrequent or no check-points is denoted as a *stake-bleeding* attack [154]. *Countermeasures:* One mitigation is to lock the deposit for a longer time than the period of participation in the consensus [25]. The next mitigation technique is frequent periodic check-pointing, which causes the irreversibility of the blockchain with respect to the last checkpoint. Another option is to apply key-evolving cryptography [147] and forward-secure digital signatures [29], which require users to evolve their private keys, while already used keys are erased [115]. Hence, signatures cannot be forged in the case of compromise. The third mitigation technique is enforcing a chain density in a time-domain [154] for the protocols where the expected number of participants in each round is known (e.g., [211]). The last mitigation technique is context-sensitive transactions, which put the hash of a recent valid block into a transaction itself [154].

VII. Replicated State Machine Layer

This layer is responsible for the interpretation and execution of transactions that are already ordered by the consensus layer (see Section VI). Concerning security threats for this layer are related to the privacy of users, privacy and confidentiality of data, and smart contract-specific bugs. We split the RSM layer into two parts. The first part deals with transactions, which are digitally signed messages by arbitrary type of an involved party (see Figure 1), and whose content will be added into the blockchain upon successful checks and sufficient fee provided – this results in an updated state of the blockchain. The second part represents a special type of transactions that are intended for the execution of pre-specified programming logic, referred to as smart contracts, whose correct execution is enforced by the blockchain. Smart contracts involve two special types of

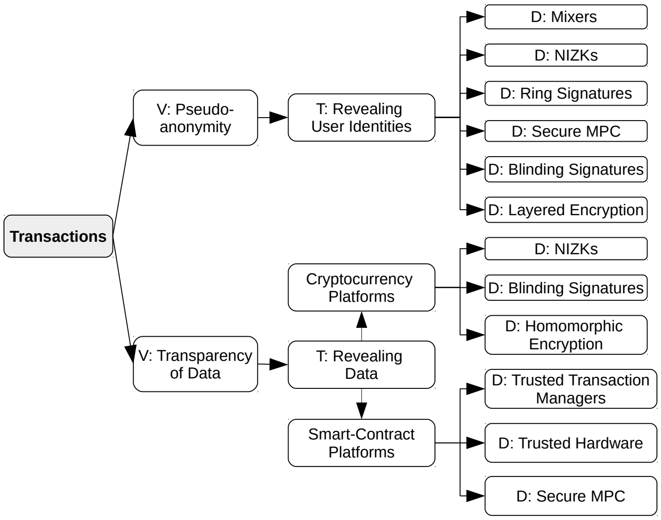

Figure 12: Vulnerabilities, threats, and defenses of privacy threats (RSM layer).

transactions, which represent the specification of programming code itself and the invocations of this code together with input data.

A. Transaction Protection

Mostly, transactions containing plain-text data are digitally signed by private keys of users [265], [139], enabling anybody to verify the validity of transactions with the corresponding public keys. However, such an approach provides only pseudonymous identities that can be traced to real IP addresses (and sometimes to identities) by a network-eavesdropping adversary, and moreover, it does not ensure confidentiality of data [143]. Therefore, several blockchain-embedded mechanisms for privacy of data and user identities were proposed in the literature, which we further elaborate on. Note that privacy preserving techniques can be applied also on the application layer of our stacked model but imposing higher programming overhead and costs (e.g., see Section IX-A, Section IX-E, and Section IX-F). This is common in the case of blockchain platforms that do not support them natively.

1) Security Threats and Countermeasures: We present a taxonomy of vulnerabilities, threats, and defenses related to privacy of transaction data and user identities in Figure 12.

Privacy Threats to User Identity. In most of the blockchains, user identities can be linked with their transactions by various deanonymization techniques, such as network flow analysis, address clustering, or transaction fingerprinting [143], [41], [305]. Moreover, blockchains designed with anonymity and privacy features (e.g., Zcash, Monero) are also vulnerable to a few attack strategies [205], [260]. *Countermeasures:* Various means are used for obfuscating user identities, including centralized [246], [59] and decentralized [319], [43], [402] mixing services, ring signatures [371], [274], and non-interactive zero-knowledge proofs (NIZKs) [324]. Some mixers enable internal linkability by involved parties [246] or linkability by the mixers [59], which are also potential threats. Unlinkability for all parties can be achieved by multi-party computation [402], blinding signatures [370], or layered encryption [319]. An example is presented by Heilman et al. [176], where the authors utilize anonymous vouchers issued by untrusted intermediaries (in exchange for native crypto-tokens) and blinding signatures to achieve unlinkability in the setting similar to mixing services. To further optimize costs, the authors propose another scheme leveraging micro payment networks for off-chaining the transfers and voucher exchanges/redemptions; however, this scheme does not achieve anonymity against malicious intermediaries. Ring signatures [314] provide unlinkability to users in a signing group [274], [371], enabling only the verification of correctness of a signature, without revealing an identity of a signer.

Privacy of data. NIZKs [324], [133] and blind signatures [176], [370] can be used for the preservation of data privacy. Another method is homomorphic encryption [288], [293], which enables the computation of certain operations over encrypted messages (e.g., ElGamal encryption provides additive homomorphism). Privacy and confidentiality for smart contract platforms can be achieved through trusted transaction managers [221], trusted hardware [96], and secure multi-party computations [406] embedded into these platforms. Privacy of data can be achieved even on blockchain platforms without embedded support of privacy-preserving constructs. For example, Zether [71] is built on top of the public smart contract platform Ethereum, and it provides a confidential payment mechanism that embeds balance of users into (secret) exponents of ElGamal encryption. Other similar examples that deal with privacy of data at the application layer of our stacked model are presented in Section IX-A, Section IX-F, Section IX-E.

B. Smart Contracts

Smart contracts, introduced to automate legal contracts [350], now serve as a method for building decentralized applications on blockchains. They are usually written in a blockchain-specific programming language that may be Turing-complete (i.e., contain arbitrary programming logic) or only serve for limited purposes. In the following, we describe these two contrasting types of smart contract languages and their security aspects.

1) Security Threats and Countermeasures: We present a taxonomy of vulnerabilities, threats, and defenses inherent to smart contract platforms in Figure 13.

Turing-Complete Languages. An important aspect of this smart contract language category is a large attack surface due to the possibility of an arbitrary programming logic. Examples of this category are Serpent Vyper, Yul, Flint, LLL, and Solidity, while as of now Solidity is the most popular and widely-used one. *Serpent* [141] is a high-level language that was designed to be simple and similar to the Python language. However, Serpent was designed in an untyped fashion, lacking out-of-bound access checks of arrays and accepting invalid code by

Acronym / Incident	Threat / Vulnerability	Reference	Description	Impact ($)	SWC Entry
DAO Attack (June 2016)	Reentrancy	[197]	A vulnerability in the code allowed a repeated withdrawal of ether, which could be exploited with a malicious fallback function.	70,000,000	SWC-107
King of the Ether Throne (February 2016)	Unchecked return value	[126]	An unchecked return value prevented the rightful compensation of the user.	300	SWC-104
RockPaperScissors	Storing secrets	[72]	A smart contract relied on a secret value that should not be visible to other users.	—	
	Integer overflow and underflow	[281]	The user's input was not properly sanitized and overflow / underflow was possible.		SWC-101
	Unexpected value received	[242]	A smart contract may contain conditions that rely on a certain balance, but the attacker is able to send an unexpected monetary value to the contract and disrupt the intended functionality.	—	SWC-132
	Delegatecall	[243]	Solidity has a feature called delegatecall that enables remote calls of other contracts. Such calls cause an execution of untrusted contracts in the context of the caller, hence all vulnerabilities of the remote code can be exploited.	—	SWC-112
Parity multisig hack (July 2017)	Default function visibility	[357]	Solidity functions are public per default. This can be easily exploited if a developer forgets to make critical functions private.	30,000,000	SWC-100
TheRun (April 2016)	Weak source of randomness	[374]	Relying on the block number or timestamp as a source for randomness is not safe, since it can allow a prediction of the next random number.	—	SWC-120
GovernMental (March 2016)	DoS	[186]	The gas limit or failing calls to external functions can lead to situations where a contract becomes unusable.	11,000	SWC-113, SWC-128
EXTCODESIZE DoS attack (September 2016)	DoS	[376]	The attack exploited a mismatch between the computational cost of some operations and their gas cost.	—	
Rubixi (March 2016)	Unprotected Ether withdrawal	[243]	The access to administration or initialization functions had not been adjusted properly. Missing modifiers or an improperly named function might lead to unauthorized access.	—	SWC-105, SWC-118
Bancor (June 2017)	Front-running / transaction order dependency	[243]	Some contracts rely on the transaction order to make a decision. For example, a winner of a game is chosen from the first transaction with a correct answer. Since the transactions can be inspected by the consensus nodes before they are included, the attacker might steal the answer and make a transaction with a higher fee, which will be included as the first one.	—	SWC-114
	Timestamp dependency	[3]	Some contracts might use the current timestamp to trigger certain events. However, timestamps can be adjusted by malicious consensus nodes.	—	SWC-116
	Write to an arbitrary storage location	[169]	By taking advantage of neighboring addresses in the storage and un-sanitized code, the unauthorized attacker might write to sensitive storage locations.	—	SWC-124
Parity multisig hack 2 (November 2017)	Unprotected selfdestruct call	[5]	The unprotected self destruct functionality can be exploited to destroy contracts (or libraries), and potentially freeze funds in them.	150,000,000	SWC-106

Table IV: Incidents and possible vulnerabilities present at the RSM layer.

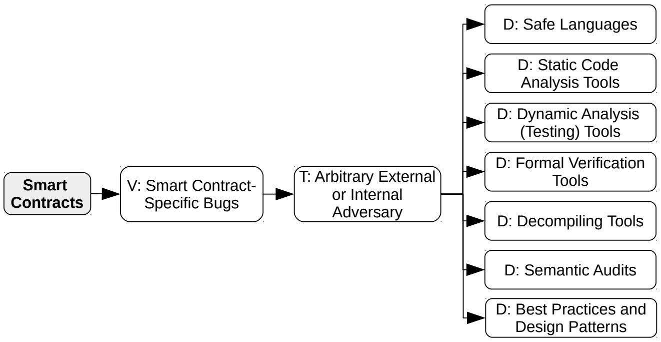
Smart Contracts
V: Smart Contract-Specific Bugs
T: Arbitrary External or Internal Adversary
D: Safe Languages
D: Static Code Analysis Tools
D: Dynamic Analysis (Testing) Tools
D: Formal Verification Tools
D: Decompiling Tools
D: Semantic Audits
D: Best Practices and Design Patterns

Figure 13: Vulnerabilities, threats, and defenses of smart contract platforms (RSM layer).

compilers [392], which opened the door for plenty of vulnerabilities. Hence, Serpent showed to be an unsuccessful attempt to simplify the coding phase. *Vyper* [74] is an experimental language designed to ease the audit of smart contracts and increase security – it contains strong typing and bounds/overflows checks. *Yul* [137] is a typed intermediate language for Ethereum, which can be compiled to bytecode for the EVM 1.0, EVM 1.5 and eWASM platforms. Snippets of Yul code can be inserted as an inline assembly within Solidity code to perform optimizations that are applicable for these three platforms. *Flint* [327] is a type-safe language for Ethereum smart contracts. The major focus of this langauge is its robustness, and it also provides some special features such as caller protection, which can help to produce robust contracts. *Lisp Like Language (LLL)* [136] is a low-level language that is similar to Assembler. It aims to be simple and to support the creation of clean code, and it removes the need to code the stack and jump management. Moreover, it enables a focus on the resource-constrained nature of Ethereum and allows an optimized use of the resources. *Solidity* [138] is an object-oriented statically-typed language that is primarily used by the Ethereum platform. Contracts written in Solidity can contain various types of vulnerabilities [17], [243], [3], which resulted in many incidents in the past. Table IV outlines the most prominent incidents and the associated vulnerabilities.[11] In addition, the table classifies vulnerabilities according to an existing smart contract weakness classification (SWC) registry [339].

Mitigation techniques for such vulnerabilities can be realized by applying static or dynamic analysis (testing) tools, formal verification tools, security audits, as well as respecting best practices and utilizing known design patterns [339], [362]. Tools for smart contract analysis are intended for a detection of various vulnerabilities. For the survey of such tools, we refer the reader to the associated literature [290], [106]. In the following, we give an overview of the most important tools in this category:

- *Static analysis tools* such as linters, try to find vulnerabilities by inspecting the source code. For example, SmartCheck [358], Solhint [304], Solium [123], and Slither [112] belong to this category. Another example, sCompile [91], works statically, but it also includes a dynamic component.
- *Dynamic analysis tools* seek for vulnerabilities while executing the code of smart contracts. For example, simple forms of dynamic analysis are unit testing with hand-crafted tests [202] or replay testing [174], where existing executions (or manually captured ones) are used to check if the same results can be reproduced. A more automated form of testing is fuzzing [348], which generates unexpected, undefined, random, or invalid inputs with the goal of triggering a crash or revealing defects and vulnerabilities. Fuzzers, like ContractFuzzer [198], Echidna [340], and Harvey [382] can be used for automated smart contract testing as well. Another technique of dynamic analysis is symbolic execution [214], where a program is executed with symbolic values (i.e., logical expressions) that make it possible to explore all reachable paths of a program. For this technique, there are tools like Securify [366], Manticore [280], Oyente [239], Mythril [261] and Osiris [361].
- *Formal verification tools* usually apply an abstract model or a semantic definition to check for security and/or correctness properties of smart contracts. One type of this category are semantic-based approaches that work with a semantic language specification defining the expected behavior of smart contracts. Examples of this type are FSolidM [245], and Kevm [187]. Other types of formal verification tools are semantic-based approaches that work with a behavioral model [2]. Several formal verification methods [262], [23], [271] apply abstract models (e.g., finite state machines) that define the expected states and outputs of smart contracts for a given input. The underlying models are used for checking the functional correctness or the presence of vulnerabilities (e.g., Zeus [204]). Additionally, such models can be applied for proving certain security properties [39], [167].
- *Decompiling tools.* The source code of contracts is often not public in contrast to their bytecode. For this reason, bytecode decompilers, like Erays [400], Eveem [218], or Porosity [347] can be used to (partially) reconstruct the source code of a contract. Additionally, there exist various static bytecode analyzers, like Maian [273], EthIR [7], Rattle [346], MadMax [165], Vandal [66], and automated exploit generators, like teether [222] that can be utilized to find vulnerabilities in the bytecode.

Turing-Incomplete Languages. The main pro of this category is its design-oriented goal of a small attack surface and the emphasis on safety, but it is achieved at the cost of limited expressiveness. Examples of this category are Pact, Scilla, Bitcoin Script, Ivy, and Simplic-

[11] Note to the best of our knowledge, for some vulnerabilities there are no publicly available references on incidents supporting the existence of vulnerabilities.

ity. *Pact* [299] is a declarative language intended for the Kadena blockchain and provides type inference and module-guarded tables to prevent direct access to the module. Pact is equipped with the ability to express and check properties of its programs, also leveraging satisfiability modulo theories (SMT) solvers. *Scilla* [331] is designed to achieve expressiveness and tractability while enabling formal reasoning about contract behavior. Every computation utilizes an automata-based model, and computations are realized as standalone atomic transitions that strictly terminate. Scilla enables external calls only as the last instruction of a contract, which simplifies proving safety and thus mitigates a few vulnerabilities. *Bitcoin Script* [215] is a stack-based language for the Bitcoin platform. It has limited complexity and processing requirements, and its main purpose is transaction processing. *Ivy* is a high-level declarative predicate language for the Bitcoin platform. It can be compiled to a Bitcoin script and its main advantage is its comprehensibility, which enables fast writing and an easy understanding of the code. *Simplicity* [279] is a typed functional language that works with combinators. It is equipped with (formal) denotational and operational semantics, which facilitate the estimation of the required computing resources.

VIII. Application Layer: Ecosystem Applications

The application layer mainly consists of end-user services and applications that are built on top of blockchains; therefore, the security threats are mostly specific to particular types of applications. Nevertheless, there are a few application-level categories that are often utilized by other higher-level applications. In the current section, we isolate such categories into a dedicated application-level group, also denoted as ecosystem, while we describe the rest of the applications in Section IX. We refer the reader to Figure 14 for a hierarchy of dependencies among particular application level categories, and their wrapping groups.

The group of ecosystem applications contains five categories: (1) **crypto-tokens and wallets**, (2) **exchanges**, (3) **oracles**, (4) **filesystems**, (5) **identity management**, and (6) **secure-timestamping**. While wallets ensure secure storage of secrets that are required for authentication at *all* blockchains, crypto-tokens (protected by wallets) usually exist in *public* blockchains, and they mostly serve to incentivize participation in the consensus protocol but also for other purposes (see Section VIII-A). For application users, it is important to securely trade with various crypto-tokens from the same blockchain or between different blockchains, which is the focus of the exchange category (see Section VIII-B). Since the blockchain is an isolated environment, for many applications it is crucial to securely deliver data from the outside world, which is the role of the oracles category (see Section VIII-C). Further, many blockchain applications require persistent storage of the data, which in some cases might be overly space-consuming, and thus native storage at the blockchain would be prohibitively expensive. Therefore, the filesystems category (see Section VIII-D) deals with these issues and also provides alternatives to native storage of data on the fully-replicated blockchains themselves. Many blockchain applications require assigned (and sometimes verified) names/identities of entities with their corresponding public keys. This problem is dealt with in the identity management category (see Section VIII-E). Finally, some application might require to timestamp data or work with already timestamped data. The blockchain itself is capable to timestamp arbitrary data by storing its hash, which is the subject of the secure timestamping category (see Section VIII-F).

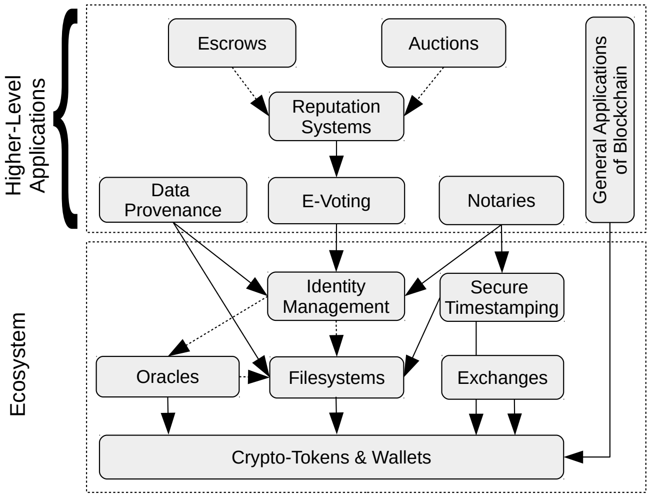

Figure 14: Hierarchy in inheritance of security aspects across categories of the application layer. Dotted arrows represent application-specific and optional dependencies.

It is worth to note that although the application layer of the reference architecture is placed on top of other layers, we emphasize that some applications require that parts of their functionality are embedded into lower layers as well. For example, decentralized filesystems (see Section VIII-D) might combine data storage as an application-layer service with the proof-of-storage consensus algorithm present at the consensus layer. Further, we note that since all applications inherently utilize authentication that is based on cryptographic constructs, operational security threats (such as secure storage of secrets) must be considered as well: we include them as part of the crypto-tokens and wallets category (see Section VIII-A). To accompany the application layer with several evidences of incidents occurred in practice, we provide the reader with Table II, which mostly describes incidents related to centralized components utilized in blockchain's infrastructure.

A. Crypto-Tokens & Wallets

Besides blockchains that provide cryptocurrencies with native crypto-tokens, there are other blockchain applications that use crypto-tokens for the purpose of providing owners with rights against a third party (i.e., counter-party tokens) or with the possibility of transferring asset ownership (i.e., ownership / colored tokens) [259]. All types of tokens require that private keys and secrets linked with user accounts are protected. For this purpose, two main categories of wallets have emerged –

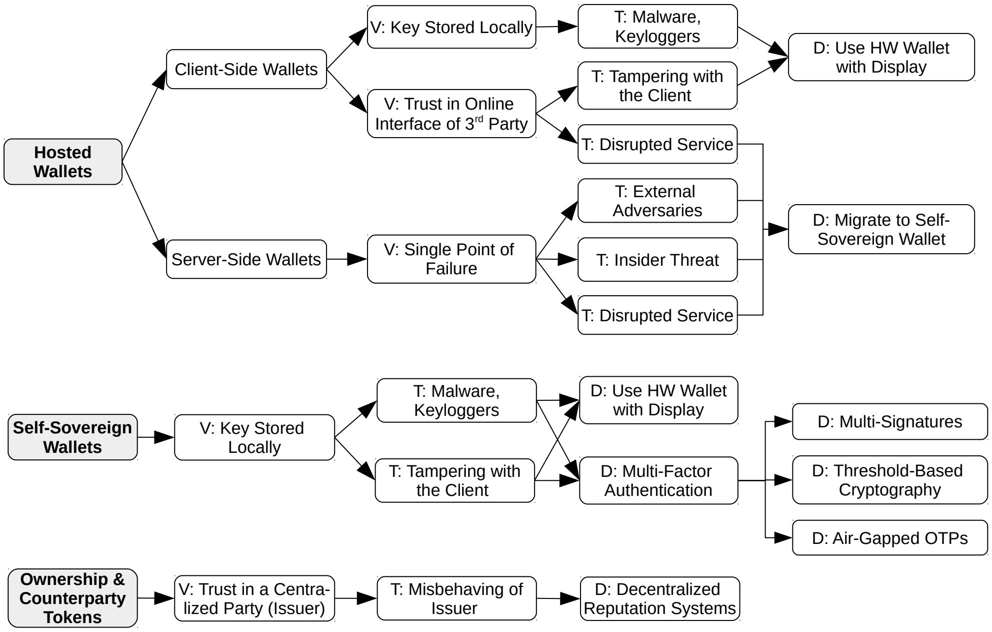
Hosted Wallets
Client-Side Wallets
V: Key Stored Locally
T: Malware, Keyloggers
D: Use HW Wallet with Display
V: Trust in Online Interface of 3rd Party
T: Tampering with the Client
T: Disrupted Service
Server-Side Wallets
V: Single Point of Failure
T: External Adversaries
T: Insider Threat
T: Disrupted Service
D: Migrate to Self-Sovereign Wallet
Self-Sovereign Wallets
V: Key Stored Locally
T: Malware, Keyloggers
T: Tampering with the Client
D: Use HW Wallet with Display
D: Multi-Factor Authentication
D: Multi-Signatures
D: Threshold-Based Cryptography
D: Air-Gapped OTPs
Ownership & Counterparty Tokens
V: Trust in a Centralized Party (Issuer)
T: Misbehaving of Issuer
D: Decentralized Reputation Systems

Figure 15: Vulnerabilities, threats, and defenses of the crypto-token & wallets category (application layer).

self-sovereign wallets[12] and *hosted wallets* [132], [58], [188]. All crypto-tokens are exposed to technical and regulatory risks, while non-native tokens are in addition exposed to legal risks [259]. In the following, we describe two main categories of key management approaches and security implications for their usage.

Self-Sovereign Wallets. Users of self-sovereign wallets locally store their private keys and directly interact with the blockchain platform using the keys to sign transactions. The instances of these wallets differ in several points. One of them is the manner in which the keys are isolated – there are software wallets that store the keys within the user PC (e.g., Bitcoin Core [45], Electrum Wallet [127], MyEtherWallet [264]) as well as hardware wallets that store keys in a sealed storage, while they expose only signing functionality (e.g., Trezor [364], Ledger [227], KeepKey [207], BitLox [52], CoolBitX [109], Ellipal [128]). Another type of wallets enables functionality and security customization through a smart contract (e.g., TrezorMultisig2of3 [368], Ethereum MultiSigWallet [107], SmartOTPs [188]).

Hosted Wallets. Hosted wallets require a centralized party to provide an interface for interaction with the wallet and thus the blockchain. If a hosted wallet has full control over private keys, it is referred to as a *server-side wallet* (e.g., Coinbase [103], Circle Pay Wallet [99], Luno Wallet [238]), while in the case of keys stored in the user's browser, the wallets are referred to as *client-side wallets* (e.g., Blockchain Wallet [54], BTC Wallet [68], Mycelium Wallet [263], CarbonWallet [86], Citowise Wallet [100]). We refer the reader to works [188], [132] for a security overview of miscellaneous wallet solutions.

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the crypto-token wallets category in Figure 15. Server-side wallets present a single point-of-failure, which can be exploited by external or internal adversaries, and moreover it can be subjected to availability attacks such as DoS. Since server-side wallets have been the target in several security incidents [379], [307], [312], their popularity has declined in favor of client-side wallets. Client-side wallets do not expose private keys to a centralized party but store it locally. Nevertheless, they still trust in the online interface provided by such a party, and moreover, their availability is dependent on this party. Other threats with client-side wallets are client tampering and malware/keyloggers, which focus on deceiving the user while signing a transaction and stealing the key, respectively. Possible mitigations of these attacks include hardware wallets that display details of transactions to the user, while the user confirms signing by a button (e.g., Trezor, Ledger, KeepKey).

By contrast, self-sovereign wallets do not trust in a third party nor rely on its availability. However, these wallets are susceptible to key theft (i.e., malware [118], keyloggers [58], [295]). One protection is to use a hardware wallet with a display as described above. Another option is to protect self-sovereign wallets by multi-factor/(-step) authentication using multi-signatures [368], [107], threshold-based cryptography [162], or air-gapped OTPs [188]. In the case of counter-party and ownership tokens realized at the application layer of existing public blockchains, we emphasize the additional vulnerability caused by trusting in the centralized party that issues such tokens – the provided counter-party and ownership rights are only virtual, which imposes a significant risk. While this risk cannot be eliminated in this application scenario, a possible mitigation technique for preventing fraudulent issuers is to use decentralized reputation-based systems (see Section IX-B) and notaries (see Section IX-D) that might built on top of them.

[12] Note that self-sovereign wallets are also referred to as non-custodial wallets.

B. Exchanges

If the user (with any node type) wishes to exchange crypto-tokens or colored tokens of one blockchain for crypto-tokens of another blockchain, she might either find a counter-party wishing to exchange the opposite pair or approach a decentralized exchange (DEX).[13] In the first case, the *atomic swap protocol* [47] handling an exchange between two parties can be applied directly, while in the second case, DEX might serve as the counter-party or intermediary (especially for colored token exchanges). When DEX serves as an intermediary, the atomic swap protocol can be extended to involve three parties (a.k.a., *three-way atomic swap* [180]). Furthermore, exchanges of tokens might occur even within the same blockchain. We describe all these cases in the following.

Cross-Chain Atomic Swaps. Atomic swaps assume two parties $\mathbb{A}$ and $\mathbb{B}$ owning crypto-tokens in two different blockchains. $\mathbb{A}$ and $\mathbb{B}$ wish to execute cross-chain exchange atomically and thus achieve a *fairness* property, i.e., either both of the parties receive the agreed amount of crypto-tokens or neither of them. First, this process involves an agreement on the amount and exchange rate, and second, the execution of the exchange itself.

The atomic swap protocol [275], [47] enables conditional redemption of the funds in the first blockchain to $\mathbb{B}$ upon revealing of the hash pre-image (i.e., secret) that redeems the funds on the second blockchain to $\mathbb{A}$. The atomic swap protocol is based on two Hashed Time-Lock Contracts (HTLC) that are deployed by both parties in both blockchains. Although HTLCs can be realized by Truing-incomplete smart contracts with support for hash-locks and time-locks, for clarity, we provide a description assuming Turing-complete smart contracts:

1) $\mathbb{A}$ chooses a random string x (i.e., a secret) and computes its hash $h(x)$. Using $h(x)$, $\mathbb{A}$ deploys $HTLC_{\mathbb{A}}$ on the first blockchain and sends the agreed amount to it, which later enables anybody to do a conditional transfer of that amount to $\mathbb{B}$ upon calling a particular method of $HTLC_{\mathbb{A}}$ with $x = h(x)$ as an argument (i.e., hash-lock). Moreover, $\mathbb{A}$ defines a time-lock, which, when expired, allows $\mathbb{A}$ to recover funds into her address by calling a dedicated method: this is to prevent aborting of the protocol by the another party.
2) Upon $\mathbb{B}$ notices that $HTLC_{\mathbb{A}}$ has been already deployed, she deploys $HTLC_{\mathbb{B}}$ on the second blockchain and sends the agreed amount there, enabling a conditional transfer of that amount to $\mathbb{A}$ upon revealing the correct pre-image of $h(x)$ ($h(x)$ is visible from already deployed $HTLC_{\mathbb{A}}$). $\mathbb{B}$ also defines a time-lock in $HTLC_{\mathbb{B}}$ to handle abortion by $\mathbb{A}$.
3) Upon $\mathbb{A}$ notices deployed $HTLC_{\mathbb{B}}$, she calls a method of $HTLC_{\mathbb{B}}$ with revealed x, and in turn, she obtains the funds on the second blockchain.
4) Upon $\mathbb{B}$ notices that x was revealed by $\mathbb{A}$ on the second blockchain, she calls a method of $HTLC_{\mathbb{A}}$ with x as an argument, and in turn, she obtains the funds on the first blockchain.

If any of the parties aborts, the counter-party waits until the time-lock and redeems the funds. The previous description assumes an on-chain atomic swap; however, off-chain atomic swaps are also possible, and they provide an almost instantaneous response, e.g., off-chain swaps are possible in two-parties payment channels and their extension to multiple parties known as the lightning network [298].

Cross-Chain Decentralized Exchange. Although atomic swaps are, in theory, sufficient means for execution of an exchange between two parties on different blockchains, the situation is more difficult in practice. In particular, there might not exist a contra-party wishing to exchange the opposite pair, or the user is not aware of it. This opens the door to decentralized crypto-token exchanges (DEX), which might facilitate the process of maintaining and matching the existing orders, act as a contra-party or an intermediary, while still guaranteeing the fairness. Some DEXes maintain a list of buy & sell orders, where each order contains a fee for an exchange, rewarding DEX for its service. The users match the orders, reward DEX, and afterwards perform atomic swap on their own. The first DEX that started to support atomic swaps is Komodo [219].

If users wish to trade more obscure crypto-tokens, for which there is no matching counter-order, DEX may serve as a counter-party and do the atomic swap with the user. Moreover, if the users wish to trade colored tokens for native crypto-tokens of difference blockchains (e.g., $\mathbb{A}$ intends to sell an asset for BTC, while $\mathbb{B}$ intends to buy the asset for ETH), DEX might serve as an intermediary who facilitates the exchange. In detail, DEX can execute the atomic swap protocol two times, once with each party (yielding eight transactions), or DEX can perform the three-way atomic swap [180] (yielding six transactions). In the following, we outline three-way atomic swap protocol, where party $\mathbb{A}$ wishes to sell asset a for BTC, party $\mathbb{B}$ wishes to buy a for ETH, and DEX $\mathbb{E}$ is intermediating the asset transfer:

1) $\mathbb{B}$ chooses a random string x (i.e., a secret) and computes its hash $h(x)$. Using $h(x)$, $\mathbb{B}$ deploys $HTLC_{\mathbb{B}}$ on the Ethereum blockchain and sends the agreed ETH amount there, which later enables anybody to do a conditional transfer of that amount to $\mathbb{E}$ upon calling a particular method of $HTLC_{\mathbb{B}}$ with $x = h(x)$ as an argument. Moreover, $\mathbb{B}$ defines a time-lock to handle abortion by any party.
2) Upon $\mathbb{E}$ notices that $HTLC_{\mathbb{B}}$ has been already deployed on the Ethereum blockchain, she deploys $HTLC_{\mathbb{E}}$ on the Bitcoin blockchain and sends the agreed BTC amount there, enabling a conditional transfer of that amount to $\mathbb{A}$ upon revealing the correct pre-image of $h(x)$ (which is visible in already deployed $HTLC_{\mathbb{B}}$). $\mathbb{E}$ also defines a time-lock in $HTLC_{\mathbb{E}}$.
3) Upon $\mathbb{A}$ notices that $HTLC_{\mathbb{A}}$ has been already deployed on the Bitcoin blockchain, she deploys $HTLC_{\mathbb{A}}$ on the asset blockchain and lock the asset a there, enabling a conditional transfer of a to $\mathbb{B}$ upon revealing the correct

[13] Note that centralized exchanges present another option, imposing the same security implications as server-side hosted wallets (see Section VIII-A). Therefore, we do not deal with them in the current section.

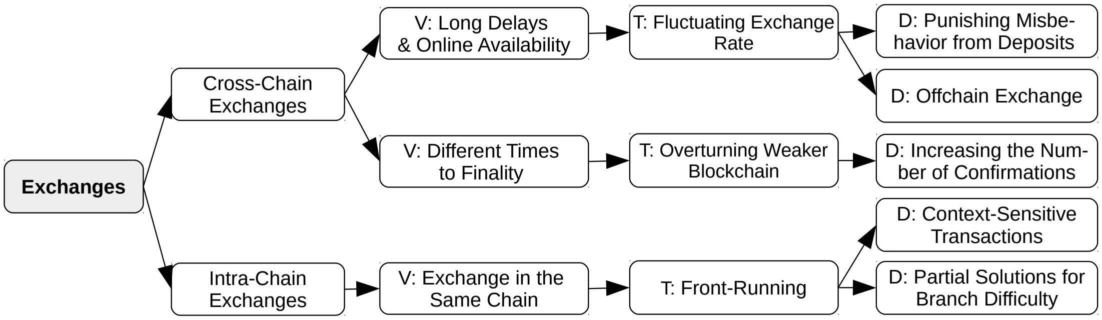
Exchanges
Cross-Chain Exchanges
Intra-Chain Exchanges
V: Long Delays & Online Availability
V: Different Times to Finality
V: Exchange in the Same Chain
T: Fluctuating Exchange Rate
T: Overturning Weaker Blockchain
T: Front-Running
D: Punishing Misbehavior from Deposits
D: Offchain Exchange
D: Increasing the Number of Confirmations
D: Context-Sensitive Transactions
D: Partial Solutions for Branch Difficulty

Figure 16: Vulnerabilities, threats, and defenses of the exchanges category (application layer).

pre-image of $h(x)$ (which is visible in already deployed $HTLC_{\mathbb{B}}$ and $HTLC_{\mathbb{E}}$). $\mathbb{A}$ also defines a time-lock in $HTLC_{\mathbb{A}}$.

4) Upon $\mathbb{B}$ notices that both $HTLC_{\mathbb{E}}$ and $HTLC_{\mathbb{A}}$ have been already correctly deployed, she reveals the secret x as a part of the transaction sent to $HTLC_{\mathbb{A}}$. This triggers a transfer of asset a to $\mathbb{B}$.
5) Upon $\mathbb{A}$ notices that x was revealed, she sends a transaction with x to $HTLC_{\mathbb{E}}$, obtaining BTC from $\mathbb{E}$.
6) Upon $\mathbb{E}$ notices that x was revealed, she sends a transaction with x to $HTLC_{\mathbb{B}}$, obtaining ETH from $\mathbb{B}$.

Tesseract [34] is an example of a decentralized real-time exchange that leverages TEE. To enter the system, users first submit time-locked refill transactions paying to Tesseract's controlled address, and then Tesseract uses its embedded lightweight SPV client to verify the inclusion of these transactions (with enough confirmations) in the target blockchain. To do exchanges, users submit bid & sell request to Tesseract, which performs matches and executed trades within the SGX enclave. When users want to sync with the on-chain state, they ask Tesseract to generate settlement transactions. The authors of Tesseract incorporate the Paxos consensus protocol among multiple mutually untrusted Tesseract nodes to increase fault-tolerance and avoid funds of users becoming stuck.

Intra-Chain Decentralized Exchange. Some intra-chain DEX designs (e.g., Maker Market, EtherOpt, and Intrinsically Tradeable Tokens [277]) require parties to post buy & sell offers on the blockchain, while smart contracts perform matches and execute trades. However, it turns out to be expensive since each placing of an order or its modification requires payment for a transaction inclusion. Therefore, designs with off-chain order matching became more popular; in these designs, only trades are executed on-chain, while orders and their matching is performed off-chain. A prominent example is 0x [375] protocol for decentralized exchange of ERC20 tokens, which utilizes Ethereum smart contracts for executing the trades. An example application of 0x protocol is EtherDelta [135].

Anoter inter-chain exchange design is know as the *automated market maker* (AAM) [37]. AAM is applicable within a smart contract-based DEX that contains deposited reserves of traded ERC20 tokens; examples are Euler [199], Uniswap [369], and Bancor [183]. AAM does not require users to match their orders, and users can proceed to exchange with the smart contract directly, which provides higher liquidity than in the case of order matching. The exchange rate is specified within the smart contract and can be automatically increased or decreased based on the demand.

Cross-Chain Communication. The concept of cross-chain exchange can be further generalized into cross-chain communication (CCC), which deals with the interoperability of applications running on different blockchains. The security aspects of CCC are very similar to the exchanges, and we refer the interested reader to the work of Zamyatin et al. [389] for a detailed description and security analysis of existing CCC designs.

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the exchanges category in Figure 16. Since intra-chain exchanges are executed in the single blockchain, they give rise to the transaction front-running, in which the adversary (a.k.a, arbitrage bots) front-run user trades by exchange transactions containing higher fees and by optimizing network latency [113]. Moreover, such adversaries might compete with each other by "biding" a higher transaction fee [113], which in turn targets the ordering mechanism of the consensus layer, where miners might tend to overturn or fork the blockchain while including only transactions with the highest fees. A mitigation technique is context-sensitive transactions [154], which do not allow overturning of the blockchain, only its extension. The same effect can be achieved by partial solutions included into branch difficulty computation [353], [390], [315], [292]. Note that context-sensitive transactions and partial solutions are a means of the consensus layer.

Since different blockchains of cross-chain exchanges might have a different time to finality, the likelihood that one blockchain will be overturned is higher than in the case of the other one. Therefore, the number of required confirmations might be agreed upon by involved parties beforehand. However, this results in longer delays for the execution of the protocol and the need for both parties to be online. In some cases, such long delays might cause fluctuation in the exchange rate and making the exchange not attractive at a later time. As a mitigation technique, off-chain exchanges (within side-chains) might be used, where each blockchain is updated only with the final transaction. Off-chain real-time exchanges might also be achieved with the use of TEE [34], under its assumed attacker model. Another mitigation for intentional delaying of the exchange by any party is to use deposit-based bonds, which will be restored only when a particular party acts timely.

C. Oracles

Oracles are trusted entities that provide plausible data that reflects the state of the world beyond the blockchain – these data are also referred to as *data feeds*. The authors of [170] and [129] define a few desired properties of oracle approaches that provide data feeds to smart contracts platforms:

Authenticity: Data feeds are authentic if they were produced by trusted content providers agreed by the consumers of the data feeds.

Integrity / Non-Equivocation: Provided data feeds should not be modified nor deleted after creation. Therefore, content providers should guarantee the correctness of the newly created data and publicly prove their consistency with the past.

Confidentiality: Sometimes, input parameters may contain confidential or private parameters. Therefore, an oracle should support such parameters and their handling.

Availability: Since the execution of dependent smart contracts relies on data feeds delivered by oracles, they need to provide a high availability (this is difficult to achieve by centralized oracles).

Easy Parsing: Data feeds that are easily parsed by smart contracts allow for straightforward audits and low processing costs.

Easy Adoption and Deployment: All involved parties should be able to start using the oracle service without substantial changes to the underlying infrastructure and the protocols used.

Prediction markets (e.g., Augur [294], Gnosis [159]) were created for the purpose of trading the outcome of events – individuals are incentivized to accurately wager on outcomes, while these outcomes serve as data feeds. *Dedicated data feeds* build on existing blockchain platforms (e.g., Oraclize [302], Town Crier [393], PDFS [170]) or create dedicated oracle networks (e.g., ChainLink [129], Witnet [116]) that internally run a consensus protocol for decentralized agreement.

Oraclize [302] enriches the data provided to smart contracts by authenticity proofs that are built upon various technologies such as auditable virtual machines[14] and trusted computing. Since authenticity proofs can be large, Oraclize can store these proofs in the distributed file system IPFS [224] instead of directly providing them to the smart contracts. To verify the authenticity proof, Oraclize provides a standalone application and an online interface called the network monitor that performs the client-side verification.

Town Crier [393] is an approach that provides authenticated data feeds to smart contracts by bridging them with the public webs through a trusted execution environment (TEE) component. A linkage of TEE with a smart contract is made by publishing a PK of TEE to the concerned smart contract and verification of the code running in TEE is achieved by remote attestation. Town Crier relies on the X.509 public key infrastructure (PKI) for TLS certificates, thanks to which provided data are provably authenticated. It also supports encryption of the request parameters, which contain, e.g., user credentials to web sites; these parameters are encrypted by a public key of the TEE.

PDFS [170] allows content providers to link their web resources with corresponding smart contracts in the blockchain. In PDFS, the data of content providers are distributed in an auditable manner, enabling publicly-verifiable data transparency and consistency of data with the past. Besides content providers, PDFS introduces two entities that arrange a smart-contract-based agreement by specifying a particular content provider required for the execution of the code in the agreement. To ensure consistent updates with the past, the authors apply consistency proofs present in a data structure called a history tree [111]. The authors additionally support the means to publicly prove censorship by a content provider.

ChainLink [129] builds on top of existing blockchains with support for smart contracts, and it distributes the provisioning of data feeds to multiple oracle providers. In detail, ChainLink maintains oracle providers and their reputation, who are selected by a smart contract based on their reputation to form an aggregated final result. When building the final result, ChainLink discards outliers and applies the BFT protocol to reach consensus on a final aggregated value. Furthermore, ChainLink discusses freeloading attacks, where an oracle provider might copy a publicly visible value provided by other oracles without any effort – the authors propose a commitment scheme to cope with this attack. Moreover, to reduce the on-chain cost of BFT execution, the authors discuss the use of threshold-based signatures for collective off-chain signing of the final value; however, freeloading attacks remain unresolved in this case so ChainLink does not use this option. ChainLink uses SLA made by users and ChainLink, and thanks to it, only valid users are able to provide ratings within reputation systems for oracle providers.

Witnet [116] details an approach similar to ChainLink but in contrast to ChainLink, Witnet runs its own oracle network with a native token. Witnesses (i.e., content providers) earn reputation points when the content that they deliver matches with the majority's content, and they lose reputation points otherwise. The reputation points serve as a stake in the consensus protocol of the oracle network, hence the higher a node's reputation, the higher the chance that it produces a block. Since more witnesses might become block producers, Witnet allows multiple chains to exist in parallel, thus forming a DAG. Witnet also supports time-lock preconditions, which concern requests that cannot be evaluated now but only in the future.

1) Security Threats: We present a taxonomy of vulnerabilities, threats, and defenses related to the oracle category in Figure 17. The data provision time of prediction markets may be too long for many applications and the provided set of data events may also be limited. By contrast, dedicated data feeds enrich the data domain and significantly shorten the provisioning times. However, they often rely on a trusted party [170], [302], which may misbehave or accidentally produce wrong data. Oracle networks eliminate trust in a single party by group consensus; however, threats related to the consensus layer of

[14] The authors do not detail what exactly they mean by this term, so it is not clear whether the audit logs are protected by some public blockchain.

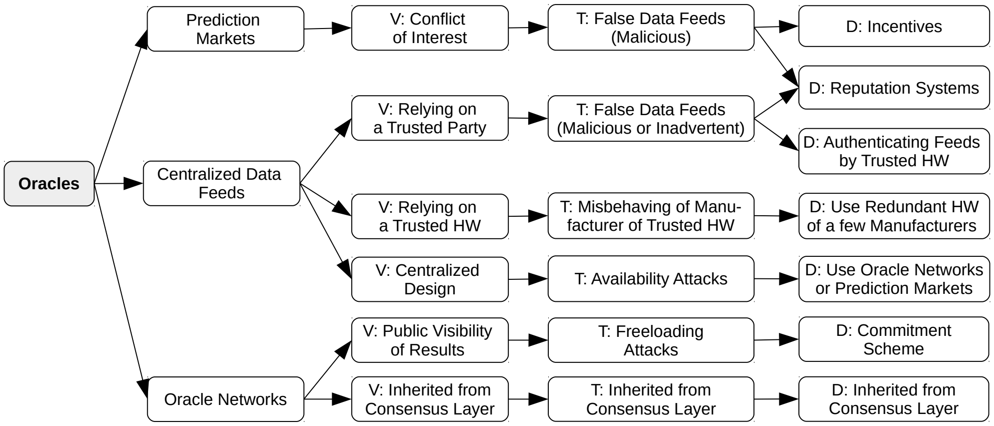
Oracles
Prediction Markets
Centralized Data Feeds
Oracle Networks
V: Conflict of Interest
V: Relying on a Trusted Party
V: Relying on a Trusted HW
V: Centralized Design
V: Public Visibility of Results
V: Inherited from Consensus Layer
T: False Data Feeds (Malicious)
T: False Data Feeds (Malicious or Inadvertent)
T: Misbehaving of Manufacturer of Trusted HW
T: Availability Attacks
T: Freeloading Attacks
T: Inherited from Consensus Layer
D: Incentives
D: Reputation Systems
D: Authenticating Feeds by Trusted HW
D: Use Redundant HW of a few Manufacturers
D: Use Oracle Networks or Prediction Markets
D: Commitment Scheme
D: Inherited from Consensus Layer

Figure 17: Vulnerabilities, threats, and defenses of the oracles category (application layer).

this functionality also need to be considered (see Section VI). Moreover, for providers that offer authenticated data feeds using trusted hardware [302], [393], a vulnerability in trusted hardware (caused by a manufacturer) may result in the entire data feed being compromised. Therefore, as a mitigation, it is possible to use several redundant instances of trusted hardware from multiple manufacturers. It is important to note that for trusted computing solutions, the assumption is that the code executed in it is bug-free, and thus one might not use return-oriented programming or other techniques to ex-filtrate sealed secrets – this is out-of-scope for the attacker model for trusted computing. Finally, centralized dedicated data feeds are vulnerable to availability attacks, such as DoS. A mitigation technique is to use solutions with a higher redundancy, such as oracle networks or prediction markets if possible.

D. Filesystems

Filesystems (FS) serve as a distributed data storage infrastructure that borrows ideas from peer-to-peer file storage systems, while it additionally incentivizes data preservation through crypto-tokens. Filesystems may or may not involve native blockchains. In the line of native blockchain approaches, a naive approach is to store the full content of the arbitrary data at the blockchain, and thus achieve full data replication. An example is storing data using the instruction OP_RETURN in Bitcoin [265] or storing data as {key:data} pairs within Namecoin [266] blockchain. However, such an approach results into a high storage overhead required for full replication of the data among the nodes: although the stored data has an extremely high durability,[15], the network expansion factor[16] is very high, which negatively affects the total costs. Therefore, the size of the data in such native blockchain approaches with full replication is usually limited.

To decrease the network expansion factor (and thus costs) while preserving a sufficiently high durability, partial replication of the data with erasure encoding is often used. In erasure encoding, the data block is encoded using two numbers (k, n), where n represents the number of total erasure shares and k represents the minimal number of erasure shares required to recover the data block. Examples that employ such an approach are Permacoin [253], Storj [378], and KopperCoin [220].

Permacoin [253] incorporates Proof-of-Retrievability in the consensus layer, where consensus nodes store large segments of data provided by an authoritative file dealer who uses the blockchain to publish Merkle roots that aggregate groups of segments. KopperCoin [220] follows a similar approach as Permacoin, but it does not need the trusted dealer for the initial distribution of static data files, as the files are selected and uploaded by the users.

Storj [378] does not use the global distributed ledger for all data transactions, since the overhead for running a distributed coordination within a consensus protocol negatively affects the throughput of the system [24], [156]. Nevertheless, the system utilizes a distributed coordination (and thus consensus) protocol for the settlement of payments. Permacoin, KopperCoin, and Storj enable probabilistic audits proving that nodes store certain data.

Audits in Proof-of-Retrievability (e.g., Permacoin, Storj, KopperCoin) only guarantee that a node possesses given data at the time of the challenge. In contrast, audits in Filecoin [303] can guarantee the data possession over a certain time range in a setting of Proof-of-Spacetime. Moreover, Filecoin uses Proof-of-Replication [32] which guarantees physically unique copies of data for each node. Filecoin utilizes zk-SNARKs and extracts randomness from the blockchain itself to run Proof-of-Spacetime and Proof-of-Replication. Filecoin can be implemented as an incentive mechanism of any blockchain or distributed storage system that allows for the verification of Filecoin's proofs (e.g., IPFS [224]).

IPFS [224] and Swarm [140] utilize the concept of distributed hash tables (DHT) [309], [345]. DHT provides a decentralized data lookup service with {key:data} mappings, in which the set of nodes storing the data is unambiguously determined by the key associated with the data (i.e., usually the hash of the data in the query). Since the lookup service and data storage are (partially) distributed, a change in the set of participants causes only a negligible disruption of availability. On the other hand, since no erasure encoding is employed in IPFS and Swarm, the network expansion factor is high,

[15] Defined as the likelihood that data will be available in the case of failures.

[16] Defined as a storage overhead for reliably storing data.

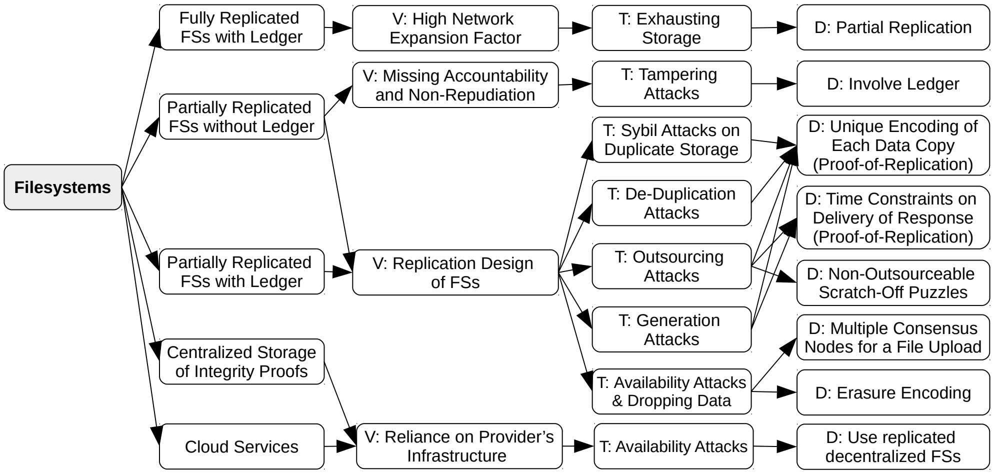
Filesystems
Fully Replicated FSs with Ledger
Partially Replicated FSs without Ledger
Partially Replicated FSs with Ledger
Centralized Storage of Integrity Proofs
Cloud Services
V: High Network Expansion Factor
V: Missing Accountability and Non-Repudiation
V: Replication Design of FSs
V: Reliance on Provider's Infrastructure
T: Exhausting Storage
T: Tampering Attacks
T: Sybil Attacks on Duplicate Storage
T: De-Duplication Attacks
T: Outsourcing Attacks
T: Generation Attacks
T: Availability Attacks & Dropping Data
T: Availability Attacks
D: Partial Replication
D: Involve Ledger
D: Unique Encoding of Each Data Copy (Proof-of-Replication)
D: Time Constraints on Delivery of Response (Proof-of-Replication)
D: Non-Outsourceable Scratch-Off Puzzles
D: Multiple Consensus Nodes for a File Upload
D: Erasure Encoding
D: Use replicated decentralized FSs

Figure 18: Vulnerabilities, threats, and defenses of the filesystems category (application layer).

which negatively influences the costs of storage. Although IPFS does not involve a blockchain, it can achieve most of its properties. In particular, nodes may optionally store a BitSwap ledger that logs data transfers with other nodes, which allows keeping track of the history and avoids tampering (i.e., by providing auditability and immutability). In contrast, Swarm is implemented on top of the Ethereum smart contract platform, and thus inherits its properties.

Alternatively to decentralized filesystems, decoupling of the data from the blockchain itself through the storage of on-chain integrity proofs (see Section VIII-F) and off-chain data is also an option; however, it introduces a single point-of-failure and thus may not provide sufficient availability. Besides centralized storage of off-chain data, cloud services (e.g., Amazon Web Services, Google Cloud, IBM) are promising approaches for decentralized yet manageable data storage, for which integrity and consistency proofs are stored on some blockchain.

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the filesystems category in Figure 18. While fully replicated and partially replicated decentralized filesystems handle **availability** using consensus-layer mechanisms, centralized storage with integrity proofs and cloud services relies on a provider's infrastructure. In a **Sybil attack** on a replicated FS, a malicious node claims storage of multiple copies of the same data. Similarly, in a **de-duplication** attack, more consensus nodes may collude to claim that each of them is storing an independent copy of the data, while only one of the nodes stores the data. These attacks can be prevented by a unique encoding of each data copy proposed in Proof-of-Replication [32], which, however, puts a higher distribution overhead on clients. In an **outsourcing attack**, a malicious consensus node claims storage of more data than it can physically store, while relying on data retrieval from outsourced data providers. In a **generation attack**, a malicious node can re-generate the previously uploaded data upon request using some algorithm, which may increase its chances to be rewarded: a node might commit to store a huge volume of generated data.

On top of unique encoding of each data copy, a mitigation technique for outsourcing and generation attacks is to put time constraints on the delivery of the response by a prover, as proposed in Proof-of-Replication [32]. In detail, the function used for the encoding of data replicas must not be parallelizable (e.g., symmetric encryption in CBC mode) to mitigate generation attacks. In the case of outsourcing attacks, the time constraints must distinguish whether cloud access was made or not. A similar mitigation for outsourcing attacks is the use of non-outsourceable scratch-off puzzles [253], [254], in which the computation of the puzzle requires access to the storage in a random order, so many round-trips are incurred during one attempt of solving a puzzle. Another attack might target the reputation of a network by dropping data and its redundant copies. A simple mitigation technique is to use multiple consensus nodes for a file upload, which diminishes the chance of the attack being successful. Another mitigation is to increase the durability by erasure encoding.

E. Identity Management

Identity management usually refers to binding identities of entities to their public keys. This concept is also referred to as Public Key Infrastructure (PKI). PKI has multiple security goals [148]:

Accurate Registration: The user must be unable to register an identity that she does not own.

Identity Retention: The user must be unable to impersonate an identity already registered.

Censorship Resistance: The user must be able to register any identity that she owns.

Traditional approaches to PKI are Certificate Authorities (CAs) and decentralized peer-to-peer networks called Webs of Trust [148]. However, centralized CAs represent a single-point-of-failure, which might easily result in a loss of the identity retention due to a centralization of trust [145] or censorship during registration. Similar issues stemming from centralization are also present in Certification Transparency [226], which is a centralized approach to monitor CAs. By contrast, Webs of Trust (e.g., PGP [405]) enable users to endorse others as trustworthy by signing their public keys while the verification of trustworthiness is done by checking whether the user in query was endorsed by someone who is already trusted; hence, the system does not pose a single-point-of-failure. However, identity retention can be broken since nothing prevents multiple users from creating public keys for the same identity, and moreover, censorship might still exist.

In computer science, some have conjectured that it is highly unlikely to design an identity management system in which identifiers would be selected in a *distributed* fashion, while remaining *secure* and *human-readable*. These three properties are often referred to as Zooko's triangle [377]. However, this situation has changed with the invention of blockchains; in particular, their immutability feature (see Section III).

Namecoin [266] is a native blockchain that offers identity management, which allows for unique registration of $\{key, value\}$ mappings, and thus facilitates the decentralization of identity management. However, searching for a value associated with a key requires full storage and traversal of the blockchain, which is costly. Blockstack [8] is a similar approach as Namecoin that provides decentralized DNS, but in contrast to Namecoin, it off-chains the data storage of domain name mappings while it keeps only references to hashes of zone-files at its blockchain. Zone-files are stored off-chain and contain references to data storage locations for DNS entries that are stored off-chain as well. In this way, the scalability of the service is improved. Certcoin [148] is built on top of the Namecoin blockchain, where entities publish their public keys (PK) by posting an identity-PK pair to the Namecoin blockchain. Certcoin utilizes cryptographic accumulators, which represent a space-efficient data structure that supports the verification of set membership within the $\{ID, PK\}$ domain. In particular, to verify that a given $\{ID, PK\}$ element is the most recent, the proposed scheme imposes only a logarithmic time complexity in the number of registered users.[17] Furthermore, to speed up the PK lookup queries, the authors leverage the concept of DHT[18] which enables them to achieve a constant lookup time complexity. The next example is the Ethereum Name Service (ENS) [365], which maps human-readable domain identifiers (names) to Ethereum addresses in a similar fashion as in DNS. ENS is implemented using the Ethereum blockchain and its smart contracts, but its root domain is maintained by a multi-signature smart contract owned by "trustworthy individuals in the Ethereum community," which have superuser control over the whole system. Similarly, uPort [237] utilizes smart contracts to keep a registry of user-issued claims on the Ethereum blockchain.[19] The registry maintains a mapping of the user addresses to hashes of claims that are stored off-chain. In contrast to ENS, uPort does not provide human-readable user identifiers and discusses the possibility of using ENS as a naming layer. Smart contracts for managing identities of humans, groups, objects, and machines are also used in the ERC 725 standard [131]. In ERC 725, an identity is associated with several keys, which serves different purposes – e.g., claims by individuals, quorums of keys to act on behalf of an organization's identity, access control and privileges within an organization, etc. ShoCard [193] is another example that builds on top of existing public blockchains, but in contrast to the previous example builds a sidechain containing encrypted identity-specific data such as bio-metric data, scans of IDs/driver licenses, etc. The user may then decide to whom she wants to reveal the encrypted data. Moreover, ShoCard allows for the publishing of 3rd parties' certifications about user identities and their claims.[20] The Sovrin [359] identity network is an example of identity management in a public permissioned blockchain consisting of consensus nodes approved by Sovrin Foundation. Sovrin focuses on high throughput and low operational costs; however since it is a permissioned blockchain, there exists a chance that a request to join the network as a consensus node in censored.[21] Decentralized Identifiers (DIDs) [372] represent a new type of universally unique identifiers whose control is fully decentralized, since all roots of trust are present in the blockchain and each entity might create its own root of trust. DIDs employ the same hierarchical scheme for globally unique strings as URI. DIDs map strings to DID documents that contain various data such as public keys, endpoints of the entity, or other documents stored off-chain. In DID, only the owner can create and manage her entries and she is also able to prove the ownership of her DID entries.

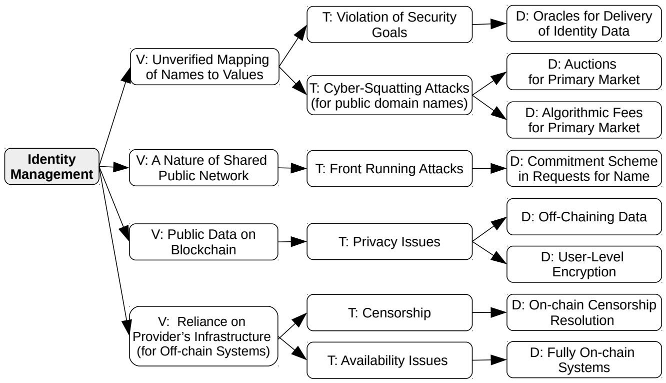
Identity Management
V: Unverified Mapping of Names to Values
T: Violation of Security Goals
D: Oracles for Delivery of Identity Data
T: Cyber-Squatting Attacks (for public domain names)
D: Auctions for Primary Market
D: Algorithmic Fees for Primary Market
V: A Nature of Shared Public Network
T: Front Running Attacks
D: Commitment Scheme in Requests for Name
V: Public Data on Blockchain
T: Privacy Issues
D: Off-Chaining Data
D: User-Level Encryption
V: Reliance on Provider's Infrastructure (for Off-chain Systems)
T: Censorship
D: On-chain Censorship Resolution
T: Availability Issues
D: Fully On-chain Systems

Figure 19: Vulnerabilities, threats, and defenses of the identity management category (application layer).

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the identity management category in Figure 19. As mentioned in work [203], most of the solutions address the problem of mapping names to values, but for the identity management system it is essential to build a mapping of entities (i.e., persons, companies) to values. However, to establish such a mapping in a trustworthy way, it would need a human arbitration or a trusted third party. For this purpose, oracles might be utilized (see Section VIII-C) to deliver verified data feeds about identity of users.

Since the space of human-readable IDs is scarce, they hold some market value in contrast to an almost infinite number of non-human-readable IDs, such as hashes. This opens a door

[17] Note that, by contrast, Namecoin imposes a linear time complexity.

[18] See brief description in Section VIII-D

[19] Note that storing claims as such belongs to the notaries category (see Section IX-D).

[20] Note that storing claims as such belongs to the notaries category (see Section IX-D).

[21] Note that this is a specific of the network layer (see Section V).

to cyber-squatting attacks, in which anybody might seize an ID that does not belong to her and then sell the ID at an inflated price. The authors Kalodner et al. [203] found in 2015 that from around $120,000$ registered names in Namecoin, only 28 were not squatted and had human-readable content. They discussed two strategies to prevent such attacks within the primary market, in which names are issued for the first time, as opposed to the secondary market where the users might trade the names they already posses. These two strategies stand for auctions and algorithmic fees, both having their respective cons. Auctions are problematic since they may start to process a request at any time, even if some users are not available. The improvement can be to fix the time when auctions start, but this enables DoS attacks on bidders and a necessity to wait. The algorithmic approach assigns the price based on the deterministic observables, such as length of the name, rank of the *.com* domain, occurrence of human-readable words, etc. Therefore, this approach may require a data feed provider (see Section VIII-C). In contrast to human-readable identifiers, non-human readable identifiers (e.g., [372] do not suffer from this threat.

Another challenge is a front-running attack (i.e., a MITM attack), in which an adversary may intercept and override the user's transaction with a malicious transaction containing the same domain but a higher fee. A prevention technique is a variant of the commitment scheme where the user first publishes a sealed (domain) name and public bid, and in the second step, submits the plain text of the name.

Further, identity-related user data that are published on the blockchain are subject to certain privacy issues. A mitigation is to keep only integrity information (such as hashes) on the blockchain, while data should be stored off-chain [193], [8], [237] and encrypted by the user's private key [193]. Moreover, all the approaches that rely on off-chain storage and service provisioning are vulnerable to availability issues and censorship attacks. A mitigation that provides censorship evidence is on-chain censorship resolution, similar to its use in [351], [170].

F. Secure Timestamping

The role of secure timestamping is to prove that some data existed prior to some point in time – also referred to as proof-of-existence. In the decentralized setting of blockchains, the blockchain serves as a trusted notary that enables such proofs since it provides immutability of the history. Nevertheless, the blockchain does not understand the semantics of data that are timestamped, and thus it cannot vet or certify them.

The simplest examples of secure timestamping are CommitCoin [101], STAMPD [344], Bitcoin.com Notary [48], and OriginStamp [157], which enable posting of a document's hash into a single blockchain transaction. OpenTimestamps [285] and POEX.IO [296] are examples that define a set of operations for creating timestamps and their verification as part of a Merkle tree that aggregates hashes of timestamped objects. The root of the Merkle tree is then stored in the blockchain and later used for verification of timestamped data.

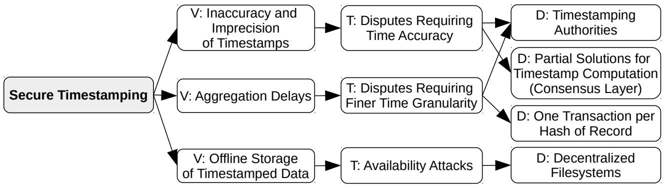
Secure Timestamping
V: Inaccuracy and Imprecision of Timestamps
V: Aggregation Delays
V: Offline Storage of Timestamped Data
T: Disputes Requiring Time Accuracy
T: Disputes Requiring Finer Time Granularity
T: Availability Attacks
D: Timestamping Authorities
D: Partial Solutions for Timestamp Computation (Consensus Layer)
D: One Transaction per Hash of Record
D: Decentralized Filesystems

Figure 20: Vulnerabilities, threats, and defenses of the secure timestamping category (application layer).

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to secure timestamping systems in Figure 20. Since secure timestamping approaches have a narrow principle of operation and provided functionality, their attack surface is very limited, too. The main security threat is inaccuracy and imprecision of timestamps provided by blockchain network as well as aggregation delays of certain secure timestamping services. A possible mitigation technique to improve accuracy of timestamps is to use timestamping authorities [352] or partial solutions for block timestamp computation [353].[22] A mitigation technique for long aggregation delays is to employ timestamping authorities or alternatively use one transaction per hash of the timestamped record, which is, however, a more expensive solution. Another class of attacks concerns the availability of timestamped data, for which decentralized filesystems might be utilized as a mitigation technique, while storing data in encrypted or plaintext form (depending on the use case).

IX. Application Layer: Higher-Level Applications

Here we elaborate on some more specific higher-level applications and use cases as opposed to ecosystem applications. We present a functionality-oriented categorization of the applications running on or utilizing the blockchain. In this categorization, we divide the applications into categories according to the main functionality/goal that is to be achieved by using the blockchain. It is important to highlight that the proposed functionality-oriented categorization is also applicable on the ecosystem group (see Section VIII) of the application layer.

In the current group of applications, we present the following categories: (1) **e-voting**, (2) **reputation systems**, (3) **data provenance**, (4) **notaries**, (5) **escrows**, (6) **auctions**, and (7) **general application of blockchains**. In the following, we will describe each of the categories, present several examples, and then summarize the potential security vulnerabilities, threats, and defenses.

We limit our description only to several most common application classes. For other detailed reviews of blockchain applications, we refer the reader to Casino et al. [88] and Zheng et al. [399], which, in contrast to our paper, apply application-oriented/domain-oriented classification.

A. E-Voting

Traditionally, voting has been paper-based by nature of its physical security attributes and entropy provided by the

[22] Note that this is a countermeasure specific to the consensus layer of the SRA.

polling booths. In such physical voting, revelations of partial tallies and vote manipulation are avoided by placing the ballot under lock. Moreover, in physical voting, the lack of receipts prevents voter coercion and vote selling. Kiayias et al. [212] and Groth [168] state several properties that are desirable in voting applications:

Perfect Ballot Secrecy: implies that a partial tally[23] can be computed prematurely only if all remaining voters are involved in its recovery.

Fairness: allows the voting protocol to compute the final tally only when all participants already had a chance to cast their vote. Thus, this property requires synchronization based on a timeout.

Public Verifiability: any public observer can verify that all cast votes are valid and that the final tally is correct. This is achieved by using a public bulletin board, which is in the context of distributed voting represented by the blockchain, and by zero knowledge proofs that confirm the validity of the submitted votes. A consequence of the public verifiability is *dispute-freeness*: i.e., the result of the voting is indisputable, since the validity of votes and correctness of the final tally can be verified by anybody.

Self-Tallying: once the voting stage has finished, anyone can compute the final tally. This property together with fairness ensures that the last voter is unable to compute the tally before casting her vote.

Fault Tolerance/Robustness: a voting protocol is able to recover from a fixed number of faulty voters who do not vote or whose vote is invalid.

Receipt-Freeness: a participant is unable to supply a receipt of her vote after casting the vote. The goal of receipt-freeness is to prevent vote selling and post-election coercion.

E-voting [93], [102] has tried to mimic many of the inherent security properties provided by paper voting. Based on whether the election is run by the voters themselves or by an authority, it is classified as decentralized or centralized. Although centralized approaches to voting are more robust and scalable, they require trust in a single party. In the decentralized setting, the blockchain election is carried out in phases and requires a multiparty computation [212], [168] executed by the voters. Decentralized voting involves a certain amount of interaction between participants and is less robust with respect to fault tolerance – i.e., if voters drop out midway, a recovery round has to be initiated. The main advantages of using the blockchain for e-voting are its immutability,[24] public verifiability, rules of the election enforced by the smart contract code, lower costs, and higher availability [60].

An example of a decentralized e-voting approach running on the blockchain is the Open Voting Network (OVN) [249], which utilizes blockchain as an instantiation of the public bulletin board for the first time. OVN is implemented through the use of an Ethereum smart contract, and it enables boardroom voting up to ∼50 voters with support for two candidate/vote choices. OVN requires the authority to initialize e-voting, compute multiparty keys from data submitted by voters, and reveal the result of the final tally. However, the authority is unable to influence the election outcome or compromise the privacy of voters (i.e., their cast votes). Although OVN preserves the privacy of voters and provides self-tallying, it does not provide receipt-freeness, hence users might be coerced to reveal their vote. Also, OVN uses deposit-based penalties to incentivize the authority and voters to actively participate. Other blockchain-based e-voting approaches are [397] and [233]. The work in [397] presents a distributed e-voting scheme that uses the blockchain, but the proposed protocol requires a restart of voting if any voter does not cast her vote. This enables sabotaging of the voting process by a single malicious voter. The work in [233] assumes an interactive honest verifier for the zero-knowledge proof presented; however, the verifier can select a biased challenge, which enables a collusion of verifier and the authority.

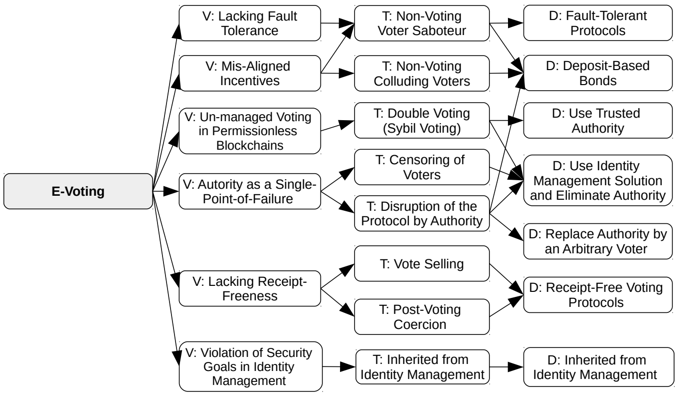
E-Voting
V: Lacking Fault Tolerance
V: Mis-Aligned Incentives
V: Un-managed Voting in Permissionless Blockchains
V: Autority as a Single-Point-of-Failure
V: Lacking Receipt-Freeness
V: Violation of Security Goals in Identity Management
T: Non-Voting Voter Saboteur
T: Non-Voting Colluding Voters
T: Double Voting (Sybil Voting)
T: Censoring of Voters
T: Disruption of the Protocol by Authority
T: Vote Selling
T: Post-Voting Coercion
T: Inherited from Identity Management
D: Fault-Tolerant Protocols
D: Deposit-Based Bonds
D: Use Trusted Authority
D: Use Identity Management Solution and Eliminate Authority
D: Replace Authority by an Arbitrary Voter
D: Receipt-Free Voting Protocols
D: Inherited from Identity Management

Figure 21: Vulnerabilities, threats, and defenses of the e-voting category (application layer).

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the e-voting category in Figure 21. The first e-voting-specific group of threats are vote selling and post-voting coercion. In vote selling, the voter can prove to a briber that she voted as agreed, while in post-election coercion the voter is coerced to show her vote by decryption of the blinded vote. A mitigation is to use receipt-free voting protocols in order to thwart both attacks [31], [27]. Existing solutions for achieving receipt-freeness assume a secret channel (bi-directional [31] or uni-directional [322]), use deniable encryption [84], [83], or employ randomizers [28].

The next threat is double-voting (with Sybil accounts) in the case of unmanaged public voting in permissionless blockchains. To prevent double-voting and ensure that only eligible voters can vote, it is usually required that a voting authority gives voters permission to vote.[25] The authority is trusted with managing voters honestly and completing all actions required for progressing the protocol from stage to stage (e.g., [249]). However, the authority represents a single-point-of-failure, since it might disrupt the execution of the protocol. Deposit-based bonds can be employed as a mitigation technique, or the authority can be replaced by an arbitrary

[23] I.e., finding partial results before the voting stage is finished.

[24] That is, a vote cannot be changed once it appears on the blockchain.

[25] This is related to know your client (KYC) compliance, and is also similar to the principles of permissioned blockchains.

voter for the execution of the protocol (but not for managing voters). Regarding the management of voters, an important threat is the authority censoring some eligible voters. A solution for eliminating the authority (or a delegation of this problem to a different application type) is to give voters permission upon successful registration of their identities within a certain identity management application (see Section VIII-E). Another issue is a possibility of (colluding) voters to not vote despite enrollment, which might result in a sabotage of the voting round (possible in e.g., in [168], [397]) or in more fine-grained inference of the actual votes of the remaining voters who voted. As in the case of a misbehaving authority, deposit-based bonds serving as penalties can be used here as well. The countermeasure for saboteurs is to use fault-tolerant voting protocols, such as [208]. Since e-voting may assume verified identities of all participants, the next group of threats and mitigation techniques is inherited from identity management (see Section VIII-E).

The last threat relates to the self-tallying property, which might not hold if no countermeasure is applied, since the last participant can compute the tally and just then decides on her vote. A simple mitigation is to use an additional "dummy" participant that is handled by the voting authority. Another solution is to enforce participants to first commit to their votes and then cast the committed votes in the later stage.

B. Reputation Systems

Reputation systems commonly serve as a means to: 1) measure the level of trust in particular entities that provide a certain service, 2) verify claims of user achievement or authenticity of issued counter-party/ownership tokens. The reputation is usually quantified based on voting of parties/users that independently analyze the history of interactions/records produced by entities in a reputation query. In a traditional centralized scenario, alongside centralization issues such as censorship, Sybil entities are also a big concern – service providers, product merchants, and competitors may participate in reputation spoofing, especially in e-commerce and its product reviews (i.e., see below bad-mouthing and ballot-stuffing attacks). Nevertheless, blockchain has a potential to address these issues.

In reputation assessment, there are two options to determine the eligibility to rate the entity in question. In the first option, an arbitrary legitimate participant can rate a product that she bought or a service that she consumed. In the second option, only a limited number of selected participants are able to vote on the authenticity of individual records (e.g., accreditation). In reputation-based systems, the identity management is two-fold, since the identity of both voters and the record owner/service provider/merchant needs to be verified.

Rating by Arbitrary Legitimate Participants. A privacy-preserving reputation system for e-commerce was proposed in [325]. The authors utilize blinding signatures and merchant-issued tokens to achieve privacy of reviews and avoid bad-mouthing and ballot-stuffing attacks. In bad-mouthing attacks, the customer (e.g., competitor) might lie about the product or service, while in the ballot-stuffing attacks, the service provider might increase her reputation by herself. A feedback-based reputation approach that utilizes the incentive-based scheme of the Bitcoin network is proposed in [85]. In this approach, any consumer might rate the service of the producer, while obtaining a voucher for the feedback. BC-MCS [398] is an example of a reputation management scheme that utilizes additive secret sharing to achieve the privacy of participants in reputation assessments.

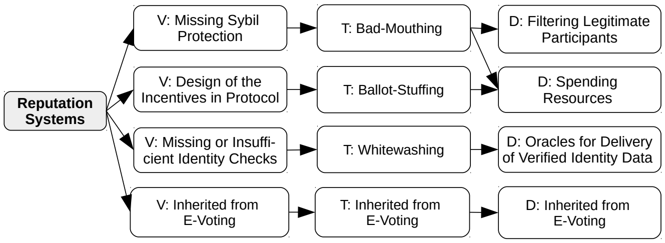
Reputation Systems
V: Missing Sybil Protection
V: Design of the Incentives in Protocol
V: Missing or Insufficient Identity Checks
V: Inherited from E-Voting
T: Bad-Mouthing
T: Ballot-Stuffing
T: Whitewashing
T: Inherited from E-Voting
D: Filtering Legitimate Participants
D: Spending Resources
D: Oracles for Delivery of Verified Identity Data
D: Inherited from E-Voting

Figure 22: Vulnerabilities, threats, and defenses of the reputation systems category (application layer).

Rating by a Number of Selected Participants. An example of such a reputation-based application is related to accreditation of educational institutions by other higher-level institutions and organizations [164].

1) Security Threats and Mitigations:

We present a taxonomy of vulnerabilities, threats, and defenses related to the reputation systems category in Figure 22. Specific security threats to reputation systems with an arbitrary number of legitimate participants are bad-mouthing, ballot-stuffing, and whitewashing attacks [325]. Bad-mouthing can be mitigated by filtering only authorized participants to submit reputation assessments (e.g., by review tokens [325], [85]). Although bad-mouthing cannot be completely prevented, it requires participants to spend resources (e.g., buying a product or paying transaction fees) in order to be eligible for rating a service provider. Similarly to bad-mouthing, the ballot-stuffing attack cannot be eliminated but only mitigated by requiring to spend resources (i.e., tokens) for each rating entry. If a service provider accumulates a significant negative reputation, it has an incentive for a whitewashing attack, in which the service provider creates a new service with neutral reputation, which is un-linkable to the previous service. To mitigate this attack, oracles for obtaining verified data about identities of entities can be employed, possibly as a part of the identity management system.

Since reputation systems in general resemble e-voting, concerning security threats are inherited from e-voting (see Section IX-A) and its dependency on identity management (Section VIII-E). In particular, only authorized participants are able to participate in the voting process and no duplicate votes are allowed (ensured by identity management), while the votes/ratings should remain blinded (ensured by e-voting) unless the particular use case requires otherwise (i.e., rating by a number of selected participants). Next, the tally of votes is required to be dispute-free, allowing all participants of the reputation system to verify its correctness.

C. Data Provenance

In general, data provenance represents the ownership history of an arbitrary object. However, in information technology, objects are represented by data that can be changed and thus the history[26] must account also for the modifications of such objects [70]. Data provenance with blockchains has a potential to resolve various disputes and issues related to intellectual property, authorship, validity of certificates, and other issued documents and intangible assets, etc. Data provenance assumes known verified identities of the involved parties (see Section VIII-E).

A potential application of data provenance is supply-chain management [223], [119], where the goal is to resolve potential issues in the traceability of goods and provenance of associated data [213]. Blockcloud is an approach that utilizes blockchains for data provenance in cloud computing [336]. The authors aim at the accountability of data creation and manipulation with the intention to detect malicious insiders and intrusions. The idea of using the blockchain for tracking packages and mails as part of supply chain management was proposed in [196]. ChainAnchor [173] is a framework for the commissioning and decommissioning of IoT devices in a cloud-based ecosystem using blockchain. In a commissioning procedure, devices prove their manufacturing provenance to a verifier (i.e., data broker) in a privacy-preserving fashion without a need for interaction with the manufacturer. Such a commission is recorded on the blockchain by a verifier. An additional goal of ChainAnchor is to incentivize owners of IoT devices to share data in a privacy-preserving manner while being rewarded. BlockVerify [1] is an anti-counterfeit solution that can be used to track the ownership of items, tracking path of goods in a supply chain, and ensuring the authenticity of particular items. A data provenance approach that focuses on the integrity of IoT-generated data is proposed in [235]. A framework to achieve data provenance of multimedia objects such as artworks and books was proposed in [40]. The authors use watermarking techniques to embed transaction metadata of objects into the objects themselves in order to prove data tampering.

Catena [360] is an approach that submits to the practical realization of data provenance by guaranteeing a non-equivocation of its append-only log and relatively low storage overhead (i.e., storing all the blockchain headers). The hash of each off-chain data block is added to the append-only log of Catena as a single transaction, which is bound to the previous transaction of the Catena's log, hence all Catena's clients see the same block history. The same method of binding consecutive transactions in an append-only log was utilized in Contour [6]. In contrast to the more general Catena, Contour focuses on non-equivocation during a distribution of open source software packages by a specified authority. In one of the proposed scenarios, the authors of [164] elaborate on the usage of blockchain for issuance of educational certificates by verified academical institutions. Such certificates might be issued by institutions whose identities are verified (see Section VIII-E), and these certificates are tamper-proof due to the immutability of blockchain.

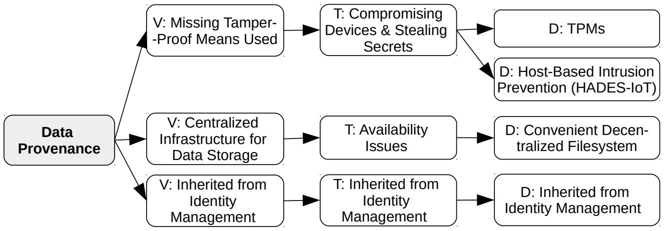
Data Provenance
V: Missing Tamper--Proof Means Used
T: Compromising Devices & Stealing Secrets
D: TPMs
D: Host-Based Intrusion Prevention (HADES-IoT)
V: Centralized Infrastructure for Data Storage
T: Availability Issues
D: Convenient Decentralized Filesystem
V: Inherited from Identity Management
T: Inherited from Identity Management
D: Inherited from Identity Management

Figure 23: Vulnerabilities, threats, and defenses of the data provenance category (application layer).

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the filesystems category in Figure 23. Since data provenance infrastructure might involve simple IoT devices and sensors, it is important to ensure that these devices are tamper-proof and no secret can be stolen from them. As a countermeasure, TPMs might be used if they are available. We note that the current trend is to involve TPMs at hardware of contemporary IoT devices. However, TPMs are often not available in a huge group of legacy IoT devices. In such cases, it is possible to leverage kernel-space and user-space memory isolation as part of intrusion prevention system, e.g., HADES-IoT [65]. Another vulnerability originates from a possible centralized design of data provenance solutions with a single data producer (e.g., [360], [6]). Here availability issues for data storage must be considered and resolved by a convenient decentralized filesystem approach. However, another vulnerability is availability of a data producer itself as well as its honesty. The assumption is that the correctness of data can be verified by clients who consume them or if a use case allows for a verification within smart contract, the auditor-free setting can be achieved.

D. Notaries

In contrast to secure timestamping, the role of the notary system is not only to prove the existence of documents at certain points in time but also to vet and certify documents [268]; hence, notary systems assume known verified identities of involved parties who do vetting (see Section VIII-E). In addition to the above two functionalities, the definition of notary system involves document preservation, which, however, in the context of the public blockchain is optional. The involved parties may decide whether to store vetted document/entries in a database of a notary service provider (e.g., [351]) or whether to keep it privately at the client side (e.g., [55]).

Blockusign [55] is an example of secure timestamping enriched by notary functionality, where notaries are represented by signers of a document that is being timestamped. Blockusign provides an off-chain interface for signing the documents, where Blockstack identity management service [8] is utilized for establishing identities of signers. In the result, only the hash of the document with digital signatures is posted to the blockchain. A blockchain-based notarization platform on Ethereum was proposed in the post [178], where an arbitrary number of users/entities with verified identities may sign/approve the documents and their new versions, respectively. The

[26] Note that sometimes the history of changes is referred to as a lineage.

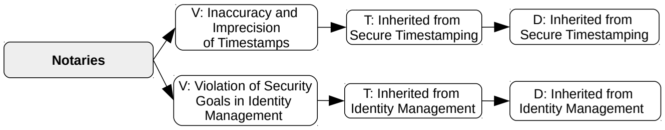
Notaries
V: Inaccuracy and Imprecision of Timestamps
T: Inherited from Secure Timestamping
D: Inherited from Secure Timestamping
V: Violation of Security Goals in Identity Management
T: Inherited from Identity Management
D: Inherited from Identity Management

Figure 24: Vulnerabilities, threats, and defenses of the notaries category (application layer).

proposal assumes a certification authority that verifies identities of involved entities, and as an example the authors suggest the use of ERC 725 standard [131]. ADVOCATE [308] is a smart contract-based approach for notarization of agreements about personal data processing in IoT ecosystems between owners of IoT devices and data processing services – both entities must co-sign an agreement to become valid. A system for property ownership is proposed in the work [258], where the author solves the buyer&seller problem by embedding a transfer of the property and the payment for it into a single blockchain transaction signed by both parties. In the proposed system, all ownership transfers can be realized without any trusted party, but the trusted party is required for introducing the initial representation record of the property and its owner onto the blockchain. The similar use case of an ownership register for vehicles was proposed [276], where trusted third parties, such as police departments, transport authorities, or workshops are used to provide and verify vehicle-specific information. SilentNotary [338] is a smart contract-based system for self-certifying of files produced by registered users the system. Note that SilentNotary has fully centralized identity management, which might be subject to tampering or censorship issues. PADVA [351] is a TLS notary service realized as a two-party agreement (i.e., SLA) implemented through smart contracts. PADVA introduces notary entities that are obligated to periodically check the state (i.e., validity) of public keys in a specified set of certificates, and in the case of state change they have to report it to the smart contract. In the case of cheating (revealed ad-hoc by clients), notaries are punished through the logic of the smart contract by loosing their deposit. Secure Host-based Event Logging (SHEL) [19] is a system that proves integrity of data produced in smart homes for the purposes of legal dispute resolution. SHEL uses forward signature scheme to sign parts of log files, while the first and the last signature associated with a log file are stored at the Bitcoin.

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the notaries category in Figure 24. Notaries inherit security threats related to timestamp accuracy from secure timestamping (see Section VIII-F). In addition, notaries must also take into account identity management security issues (see Section VIII-E), since they assume verified identities of involved parties.

E. Escrows

While blockchain-based cryptocurrencies enable native secure transfers of crypto-tokens among their owners, a challenge arises when owners want to exchange crypto-tokens they hold for assets or goods outside of the cryptocurrency blockchain. This challenge is also referred to as buyer/seller dilemma, in which the following question comes into play: "*Should the buyer trust the seller and pay her before receiving goods or should the seller trust the buyer and ship the goods before receiving the payment?*" Both parties wish to maximize fairness by executing this swap atomically and thus minimize a chance that other party will misbehave. Atomic swaps [47] resolve this problem in the context of crypto-tokens exchange between two cryptocurrencies (see Section VIII-B). However, the problem still remains in the case of trading the physical or digital goods for crypto-tokens. We note that BIP-70 [11] tried to address such trading while assuming a trusted seller in the protocol – the buyer first verifies authenticity of the seller using its X.509 certificate and then issues a transaction. The buyer might also ask the seller to interrupt the request and refund the buyer, but the seller might misbehave. [27]

In a centralized scenario, it is common to introduce a third party (e.g., eBay, Amazon, Alibaba) serving as a mediator in the case of disputes. Such an approach is also possible in the scenario with the blockchain, but there is a high chance that the mediator or the buyer will misbehave if implemented naively, while also private information about disputes/trades occurrence might leak to the public. For example, if a mediator (for seller and buyer) were to escrow a signed transaction by the buyer and release it (i.e., publish to the blockchain) only upon delivery confirmation, the buyer might perform a double-spending attack (see Section VI). Another naive example was adopted in the Silk Road marketplace [98], which operated as a Tor hidden service. In the Silk Road model, a mediator requests the sending of crypto-tokens to her address, while she is trusted to send the crypto-tokens to the seller upon delivery confirmation from the buyer.[28] However, the mediator might refuse to do it and keep the value for herself. These issues are different in contrast to a centralized scenario where fraudulent transactions can be reversed and the identities of users owning credit cards or PayPal accounts are known.

According to [161], there are two categories of fair exchange protocols that deal with *digital asset* transfer: (1) protocols transferring digital signatures that can be reconstructed and verified by a mediator, (2) protocols relying on the assumption that digital assets can be reproduced and resent by a mediator in the case of disputes. In both cases, the goal is to provide a cryptographic solution to resolve disputes. However, the authors of [161] argue that these protocols are not suitable for a trading of physical assets, where it is impossible to provide a cryptographic proof for resolution of disputes. Therefore, they propose a few protocols allowing buyers and sellers to have various combinations of properties concerning third party mediators (e.g., resistance to DoS, impossibility to send crypto-tokens to other party than buyer or seller, etc). The first type of proposed protocols contains a single mediator and involves 2-of-3 multi-signatures for splitting the control, protocols with threshold-based signatures

[27] Moreover, this protocol contained a flaw in missing authentication on the refund address, which gave rise to the Silkroad trader attack [248]

[28] Note that instead of a single mediator, the Silk Road marketplace used several temporary intermediaries with the intention of increasing the anonymity of the buyer and seller.

for improving privacy, and customized protocols leveraging additive homomorphic properties of elliptic cryptography to achieve privacy (i.e., by blinding the mediator's next address)[29] and non-interactiveness, as well as combinations of multi-signatures with bonds deposited by a mediator to avoid DoS by the mediator. Of those, the interesting protocol is called *encrypt and swap*, and it uses a specific 3-of-3 threshold signature protocol that assigns one private share to buyer and one to seller, while the third common share is known by both parties. Both parties share only their private share with the mediator, who upon dispute provide the winning party with the missing share.

In the second group of protocols, the authors propose a few group-based escrow protocols, where disputes are resolved by a majority vote of mediators. This is similar to e-voting (see Section IX-A) but in contrast to it, the identities of involved parties do not need to be verified. In these protocols, a DoS attack is thwarted as long as the majority of mediators is willing to finish the execution of the protocol. Moreover, the privacy is preserved by blinding, similarly as in the case of a single mediator protocols.

A blockchain-based distributed marketplace was proposed in [201]. The authors utilize the Ethereum smart contract platform to instantiate a marketplace contract that lists the products and an escrow agent contract that serves for resolution of disputes by a mediator. The authors discuss the usage of single mediator and group mediator protocols, as proposed in [161]. Moreover, this approach discusses the integration of logistic parties with verified identities as well as reputation-based systems to assess these parties and mediators. A practical example of a distributed marketplace is OpenBazaar [284]. OpenBazaar utilizes smart contracts to realize escrows as 2-of-3 multi-signature scheme, where the mediator is agreed by both parties, buyer and seller. An observation of OpenBazaar is that many buyers do not release the funds to the seller upon successful delivery, therefore OpenBazaar introduced a timeout, which after expiration, enables a seller to unilaterally release and acquire the funds from the escrow. Note that this scheme does not have privacy-preserving properties, and moreover it is not resistant against DoS by a mediator in the case of a dispute. On top of that, OpenBazaar creates a new contract per each Ethereum-based trade, which is not an optimal solution.

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the escrow category in Figure 25. The first group of threats refers to a misbehaving mediator who represents a single-point-of-failure. The mediator might disrupt the protocol (i.e., DoS) or decide unfairly in the case of dispute. The mitigation techniques are reputation systems for mediators, requiring a group of mediators to decide on a case or requiring the mediator to put a bond into the protocol. To avoid stealing the escrowed value by the mediator, the protocol should allow to release value only to buyer or seller. The next class of threats is related to revealing the private data about running the protocol to public/mediator as well as revealing occurrences of disputes to the public. The countermeasures are blinding mediator's address, threshold-based cryptography, including its special variant encrypt and swap. Another possible threat is double spending of an unconfirmed escrowed payment transaction by the buyer if the protocol allows it. As a prevention technique, unconfirmed transactions should not be accepted at all. Moreover, we highlight that some escrow protocols as well as atomic swaps are sensitive to double-spending performed by a selfish mining or temporary 51% attacks – therefore, in this protocols it is important to wait for enough confirmations or use blockchains with consensus protocols having a fast finality (see Section VI-A).

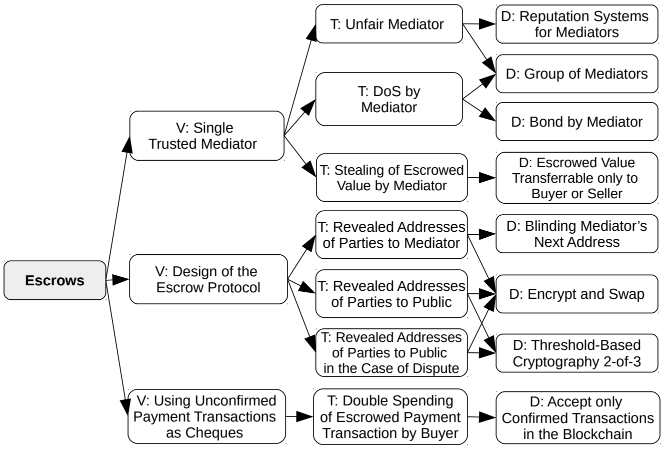
Escrows
V: Single Trusted Mediator
V: Design of the Escrow Protocol
V: Using Unconfirmed Payment Transactions as Cheques
T: Unfair Mediator
T: DoS by Mediator
T: Stealing of Escrowed Value by Mediator
T: Revealed Addresses of Parties to Mediator
T: Revealed Addresses of Parties to Public
T: Revealed Addresses of Parties to Public in the Case of Dispute
T: Double Spending of Escrowed Payment Transaction by Buyer
D: Reputation Systems for Mediators
D: Group of Mediators
D: Bond by Mediator
D: Escrowed Value Transferrable only to Buyer or Seller
D: Blinding Mediator's Next Address
D: Encrypt and Swap
D: Threshold-Based Cryptography 2-of-3
D: Accept only Confirmed Transactions in the Blockchain

Figure 25: Vulnerabilities, threats, and defenses of the escrows category (application layer).

F. Auctions

Auctions represent the mechanism where sellers promote the sale of arbitrary goods and assets while buyers place bids for them. Auctions should exhibit several attributes [151], such as privacy of bids, posterior privacy, publicly verifiable correctness, and resistance to DoS attacks. Privacy of bids ensures that values of particular bids are not revealed to any other participant nor the auctioneer before committing to them (i.e., no cheating is possible). Posterior privacy ensures that all bids remain private from the bidders, the public, and optionally auctioneer after the auction ends. Publicly verifiable correctness enables everybody to verify the results of the auction through the blockchain. Resistance against DoS ensures that no bidder or the auctioneer is able to prematurely abort an auction protocol without being penalized (this implies fair financial incentives as deposit-based bonds).

Most of following blockchain-based auction protocols mainly focus on private comparison of integers, while minimizing the interactivity of the protocol. Authors of [151] propose a sealed bid auction protocol as Ethereum smart contract that uses five stages. In the proposed protocol, bidders first commit to their sealed bids and then reveal their commitments to the auctioneer via a public key encryption. Afterward, the auctioneer deciphers the bids, determines the winner and announces it to the public. The auctioneer also has to provide zero-knowledge proofs demonstrating that each

[29] Note that such blinding enables to hide the execution of the protocol to the mediator only if no dispute has arose.

losing bid is smaller than the winning one and it fits an allowed range. Therefore, the number of transaction required is directly proportional to the number of bidders, which is a cost limitation of this scheme. This scheme provides partial privacy of bids, since the auctioneer has to reveal the winning bid and her address to the smart contract, but losing bids remain private. On the other hand, this scheme does not provide anonymity of bidders, since bidders use their existing Ethereum addresses to interact with the protocol.[30] To avoid aborting of the protocol, the authors utilize deposit-based penalties that are split to honest participants. In the case of malicious auctioneer claims that a certain commitment does not open to the secret bid, the bidder might prove the opposite to the smart contract by revealing her secret bid, causing penalization of the auctioneer. However, it leaks the bid value of the bidder. On top of it, this scheme works only with the honest auctioneer, since the auctioneer possesses all values of the bids attributable to bidders, which can be revealed to the public.[31] As their other contribution [150], the same authors improved on high costs intrinsic to their former work [151]. In detail, they utilized zk-SNARK, which requires only a single proof verification of the whole auction process, and moreover allows recycling of generated common reference string (a specific component of zk-SNARK) in the all auctions. To represent a problem of the auction in a suitable form for zk-SNARK, the authors do not use their own circuit, but utilize a general-purpose circuit generator for conversion of a code representing the auction problem to a circuit. The resulting circuit is not allowed to have loops with variable number of iterations, hence this approach requires predetermined maximal number of bidders; in the case of less bidders than the maximum, the auctioneer has to generate remaining zero-value bidders. Similarly, as in their previous work [151], the auctioneer is a single-point-of-failure who might compromise the privacy of all bids. Such a privacy issue of this approach and its successor led the same authors to propose a trusted computing-based approach called Trustee [152]. In Trustee, the bidders submit encrypted bids to an SGX enclave, which confidentially evaluate a winner and generate a transaction proving it. To avoid dropping inconvenient bids by the malicious auctioneer who controls the enclave, the authors require enclave to sign also all input bids as part of the proof. Since trusted computing and blockchain are isolated environments, the approach requires an active relay (controlled by the auctioneer) that performs transfers of messages between them, which naturally implies a vulnerability to DoS attack on availability of the relay and/or enclave. Another potential issue is the hardware failure/crash of the enclave; the authors designed an enclave as a state-full component: it generates and stores the private and public keys upon initialization. Although the author use sealed storage of this state at the hard drive, in the case of permanently failed enclave, this will not help, and hence a different solution is required – for example, allowing the auctioneer to change a public key of the enclave within the smart contract. Strain [53] is an auction protocol that guarantees privacy of bids against malicious bidders and, in contrast to [151], [150], also against the auctioneer. Strain is executed in four rounds (i.e., the authors assume four blocks), and the authors assume semi-honest (i.e., passive) auctioneer who serves as a judge verifying the correctness of zero-knowledge proofs and who must not collude with any of the bidders. In such a case, neither auctioneer nor a malicious bidder learns anything about bids of honest bidders; however the order of the bids is leaked to the public. Strain requires each bidder to publicly commit to her bid, while the proposed scheme enables a majority of honest bidders to open other bidders' commitments (in the case they abort the protocol). For the sake of efficiency (i.e., a constant number of rounds), the authors provide weaker security properties in contrast to MPC protocols, where no semi-honest judge is required. Finally, the authors propose an extension of their scheme to support the anonymity of all bidders by blinding RSA signatures and Dining Cryptographers (DC) network [92]. Another approach that preserves the privacy of the bid values (but not the privacy of their order) is proposed in [241]. The protocol requires off-chain interaction for two-party computation protocol that performs pairwise comparison of blinded bids among bidders and the auctioneer; the authors claim that protocol can be executed in three rounds of interaction with the blockchain. Each off-chain counterpart of the three steps requires $N-1$ transfers among participants, and in the case of parallel setup, $log_2(N)$ transfers.

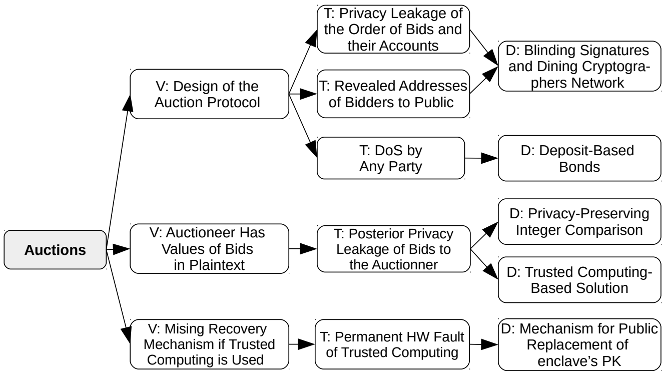
Auctions
V: Design of the Auction Protocol
V: Auctioneer Has Values of Bids in Plaintext
V: Mising Recovery Mechanism if Trusted Computing is Used
T: Privacy Leakage of the Order of Bids and their Accounts
T: Revealed Addresses of Bidders to Public
T: DoS by Any Party
T: Posterior Privacy Leakage of Bids to the Auctionner
T: Permanent HW Fault of Trusted Computing
D: Blinding Signatures and Dining Cryptographers Network
D: Deposit-Based Bonds
D: Privacy-Preserving Integer Comparison
D: Trusted Computing-Based Solution
D: Mechanism for Public Replacement of enclave's PK

Figure 26: Vulnerabilities, threats, and defenses of the auctions category (application layer).

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the auctions category in Figure 26. There are several possible issues related to privacy leakage in the auction protocols. First privacy issue stands for revealing addresses of bidders and/or order of their bids to public. A mitigation technique using blinding RSA signatures and DC network was proposed in [53]; however network-level attacks on revealing locations/IP addresses of parties remain possible. Second, in some protocols [151], [150], the auctioneer might either intentionally or accidentally (e.g., an external compromise) reveal the values of bids (including the proofs) that are attributable to particular bidders. The protection technique is to avoid the auctioneer from seeing the plaintext of the bids, and instead use privacy-preserving integer comparison or trusted computing-based solutions. In the case of trusted computing

[30] The authors discuss update of their scheme to support anonymity.

[31] Note that the authors admit this limitation in their later work [152].

Acronym / Incident	**Threat / Vulnerability**	**Reference**	**Description**	**Impact ($)**
Gatecoin (May 2016)	Centralized server compromise (single-point-of-failure)	[333]	The exchange was the victim of a man-in-the-middle attack. The malicious external party involved in this breach, and it managed to alter Gatecoin's system so that deposit transfers bypassed the multisig cold storage and went to hot wallets, which were exploited due to OPSEC issues.	25,160,000
Doge Vault (May 2014)	Centralized server compromise (single-point-of-failure)	[117]	The attacker gained access to the server where Doge Vault's virtual machines were running, providing him with full access to the systems.	56,000
BIPS.me (November 2013)	DDoS + access subversion on centralized server (single-point-of-failure)	[209]	An initial DDoS attack caused vulnerability to the system, which has then enabled the attacker to gain access and compromise several wallets.	554,260
Bitfinex (August 2016)	Centralized server (single-point-of-failure)	[185]	Although Bitfinex used 2-of-3 multisig wallets, two of these keys were owned by Bitfinex (one stored in cold wallet), while the user owned only a single key. It is not clear whether this incident involved insider threat or it was conducted externally.	71,288,416
Bitcoin Central (April 2013)	Centralized server compromise (single-point-of-failure)	[63]	Password was reset from the hosting provider's web interface, enabling the attacker to lock out of the interface and request a reboot of the machine into rescue mode. Next, the attacker has stolen private keys from the hot wallet.	—
Cyber-squatting attacks in NameCoin	Violation of accurate registration	[203]	Cyber-squatters seized identities that do not belong to them nor represent themselves.	—
Front-Running on Ethereum Exchanges	Front-running in Intra-Chain Exchanges	[113]	Arbitrage bots front-run user issued exchange transactions with the ones with the higher fees. Therefore, user issued transactions are discarded or traded with worse exchange rate as intended.	—

Table V: Incident occurred at the application layer.

solution, the assumption is that the code executed in it is bug-free, and thus one might not use return-oriented programming or other techniques to ex-filtrate sealed secrets, which is the out-of-scope attacker model for trusted computing. Another possible issue with trusting computing-based solutions is a possibility of permanent hardware failure, which requires an additional mechanism to handle (outlined above). Finally, one has to consider that a vulnerability in trusted hardware (caused by a manufacturer or its infrastructure) may result in an unfairness of the auction process.

G. General Application of Blockchains

There are many use cases of applying blockchain to the particular domain that contains mutually untrusted participants – these participants are represented by consensus nodes that execute a consensus protocol. Applications from this category focus on leveraging of inherent features provided by the blockchains and sometimes event on non-inherent ones (see Section III-B).

An example that uses permissioned blockchain for the management of healthcare data is proposed in [134], where new data are included in the blockchain upon majority agreement. The consensus nodes of this approach are represented by a patient, her family, and healthcare providers. The authors discuss an issue regarding the right to delete the personal data, which is contrary to the inherent features of the blockchain. A data protection framework for energy grids and power systems was proposed in [234]. The authors suggest smart meters act as consensus nodes and store the data of the blockchain in their memory, which are unrealistic assumptions. DistBlockNet [335] is the approach for ensuring a state consistency among multiple SDN controllers with the utilization of dedicated blockchain. In detail, flow rule tables of SDN controllers are managed by these controllers, while other entities in the system use blockchain as a reference point for downloading these tables. Another approach proposes the application of semi-permissionless blockchain in the TOR infrastructure for the purpose of more transparent updates of the list of TOR nodes [179]. A federated permissioned blockchain used as a cloud-based data storage requiring the consensus of all nodes[32] is proposed in work [149]. The authors propose to use two tiers of blockchain. The blockchain at the first tier is private and serves for consensus of participants, while the blockchain of the second tier is public PoW (e.g., Bitcoin) and serves for periodic publishing of integrity state of the first tier blockchain. Two-tier blockchain was also applied in the domain of IoT [121]. The authors of [121] deem the first tier blockchain as local to a group of IoT devices owned by a single party, while the second tier blockchain serves for

[32] Note that such a proposal has very low fault tolerance.

sharing the data among multiple untrusted parties. The authors demonstrate the applicability in a case study involving several smart homes.

1) Security Threats and Mitigations: Security threats of this category vary case by case and usually concern the privacy of data shared among involved parties [134], [234], [121]. Another common issue is an application of the blockchain with unrealistic assumptions about the target environment, e.g., low processing and storage performance of smart meters/IoT devices, missing HW support for asymmetric cryptography, tamper-proof assumptions about devices producing transactions, etc. Some works try to optimize throughput or finality of the blockchain by introducing their own consensus mechanisms (e.g., [121], [149]); however, this might not be the best option since new attack vectors might be created. A general recommendation is to study and understand security issues and countermeasures of the state-of-the-art approaches of the consensus layer (see Section VI) as well as privacy concerns present at the RSM layer (see Section VII).

X. Discussion

In this section, we outline additional security vulnerabilities and threats related to the blockchains, which, for clarity and simplicity, we have not pursued throughout our paper or mentioned them only tangentially. Further, we discuss a few types of blockchain-oriented applications that directly inherit security aspects from already existing categories; therefore, we omitted such application types in our work. In the last part, we discuss the scarce evidence of various attack types in practice.

A. Semantic Bugs

We deal with semantic bugs only at the level of the RSM layer as part of the smart contract code (see Section VII-B). However, we emphasize that semantic bugs in the blockchain infrastructure may occur at each of the proposed layers, while in the case of the RSM layer, besides smart contracts, they may occur in compilers, interpreters, etc. For the purpose of this work, we assume that the software of the blockchain-related infrastructure is implemented correctly at each of the layers and that it provides its expected functionality. On the other hand, we emphasize that these semantic bugs had already accounted for several incidents in the past, e.g., [267], [182], [46], [278]. To achieve safe and correct software at each of the layers, similarly as in the case of the RSM layer, developers and designers should utilize verification tools, testing, best practices, known design patterns, audits, etc.

B. Secure Cryptography Primitives

We emphasize that for each layer of our stacked model, we assume the use of secure cryptographic primitives with recommended key lengths [158] that are based on existing standards (e.g., [301], [90]). Examples include secure communication (i.e., network layer), the use of private keys to identify nodes in BFT protocols (i.e., consensus layer), and password management for blockchain-based services (i.e., application layer). Since the area of cryptographic primitives is standardized and extensively covered in existing research, we treat security incidents that break these primitives as out-of-scope throughout the current paper.

C. Other Blockchain-Oriented Applications

There are several other applications of blockchain that we do not mention in our work because their security aspects are inherited from one or more categories presented in Section VIII and Section IX. For example, insurance applications realized through smart contracts inherit security aspects from the oracles category, since they require data to be delivered into the blockchain from the outside world. The next example is the trading of crypto-tokens within the same blockchain platform since it has security aspects inherited from the crypto-tokens and wallets category (see Section VIII-A). Another example is the cross-chain communication, which is a generalization of the exchanges category and also inherits the most of security aspects from it.

D. Incidents in Practice

In contrast to many vulnerabilities and threats described throughout the paper, we managed to find only a few types of incidents occurred (or demonstrated) in practice (see Table II, Table III, Table IV, and Table V). However, we argue that the adoption of blockchains for practical applications is still in its infancy, and thus we may expect that the number of different incident types occurred in practice will grow.

XI. Related Work

The security reference architecture that has been presented in our work offers a comprehensive overview of blockchain-related security vulnerabilities, threats, and mitigation techniques. We adapted a custom version of the four-layer stacked model, initially presented in the work of Wang et al. [373]. In the following, we present an overview of the state-of-the-art survey papers related to blockchain research while we highlight the differences in contrast to our work. We consider three groups of blockchain-oriented research: (1) papers that use a flat categorization of threats and vulnerabilities, (2) papers that use a stacked or other multi-layered models, and (3) papers that focus on incidents that belong to a single layer.

Research with Flat Categorization. Bonneau et al. [58] present the first major survey of blockchain-specific security aspects, with a particular focus on Bitcoin and cryptocurrencies. The authors aim at the consensus-layer properties, although some network-layer aspects (e.g., DoS attacks) are discussed as well. Since smart contract functionality was that time in its early stages of development, not much is said about RSM-layer properties, and little is said about applications beyond cryptocurrencies and data storage. Similarly, Tschorsch et al. [367] and Yli-Huumo et al. [385] present early survey papers that mostly focus on consensus- and network-layer attacks, but they also deal with user privacy. The latter [385] has a particular focus on the publication details of blockchain research until 2016, e.g., the venues and the countries of the authors' institutions. Li et al. [232] present a high-level overview of blockchain security threats and incidents, but categorization is lacking. The authors deal with selfish mining, the DAO hack, BGP hijacking, and eclipse attacks, while all of them are mentioned as individual incidents. Conti et al. [108] present an overview of consensus- and network-layer attacks inherent to the Bitcoin blockchain. One interesting contribution is its overview of client-side attacks and attacks on exchange systems. Many attacks presented in this work are supported by evidence of incidents. On the other hand, the authors spent only a little effort on the issues related to the RSM and application layers.

Research with Layered or Stacked Categorization. Wang et al. [373] are the first to propose a 4-layer model denoted as "*a network implementation stack*." Despite proposing the stacked model, the authors do not focus on attacks and countermeasures concerning each of the layers. The main focus of their work is on the consensus layer, where the authors dedicate most of their attention to PoR, PoS, and BFT protocols as well as improving blockchain performance by sharding, side-chains, and non-linear data organization. In the application layer, the authors discuss a few types of emerging blockchain-based applications, such as general-purpose data storage and access control. Saad et al. [321] identify three categories of attacks: blockchain structure attacks, peer-to-peer system attacks, and application-oriented attacks. As compared to our work, it mostly holds that their *peer-to-peer system attacks* encompass our network and consensus layer attacks, whereas their *application-oriented attacks* include the RSM and application-layer attacks from our work. In contrast to our work, Saad et al. [321] put double-spending attacks into application-oriented attacks, while in our case, they are part of the consensus layer since the means for their realization resides in this layer. Moreover, the authors of this paper deal with cryptojacking attacks, which, however, are out-of-the-scope for our reference architecture, as they are not related to the infrastructure of the involved parties we consider (see Section III-A). Chen et al. [94] propose a 4-layer model similar to ours, which is used to study vulnerabilities in Ethereum. The authors identify 44 vulnerabilities, 26 attacks, and 47 defenses in total. In contrast to our model, the authors use the "*data*" layer in the place of our RSM layer. This leads to a difference in interpretation between their framework and ours: e.g., they consider re-entrancy bugs as an application layer vulnerability, whereas we treat them as an RSM-layer vulnerability. Since the authors focus on Ethereum, most vulnerabilities belong to the RSM layer; however, some of the other vulnerabilities (e.g., the BGP hijacking attack against MyEtherWallet [146], [297]) do not seem to be specific for Ethereum. In contrast, our work takes a broader view, and we do not constrain it to a single blockchain. Another stack-based model was proposed by Zhang et al. [396] and consists of six layers, where the layers stand for *the application, contract, incentive, consensus, network, and data layer*. The works of Alkhalifah et al. [10] and Zhu et al. [401] feature groupings consisting of five (*network, consensus, mining pool, smart contract* and *client* vulnerabilities) and four (*data privacy, data availability, data integrity*, and *data controllability* attacks) categories, respectively. Natoli et al. [269] strongly focus on the consensus layer but include some network-layer attacks as well (e.g., eclipse attacks and BGP hijacking).

Research Focusing on a Particular Layer. Finally, there is a number of survey papers that explicitly focus on specific layers: the network layer is analyzed in [272], the RSM layer (in particular smart contacts [17], [175], [120]), protocols of the consensus layer [80], [25], [383], [228], [153], [332], incentives at the consensus layer [20], [191], and application layer from the general standpoint [88] or with a focus on the IoT domain [289], [9], [311], [144].

XII. Conclusion

In this paper, we focused on the systematization of the knowledge about security aspects of blockchain systems, while aimed to create a standardized model for studying vulnerabilities and security threats. We proposed a stack-modeled security reference architecture consisting of four layers: (1) network layer, (2) consensus layer, (3) replicated state machine layer, and (4) application layer. At each of the layers, we surveyed options for their realization with respective security implications and properties. We modeled these options for the realization as vulnerability/threat/defense graphs, which we provided as a means for reasoning about imposed security aspects. Next, we projected the proposed reference architecture into a threat-risk assessment model ISO/IEC 15408 that captures relation among involved parties, assets, threats, and countermeasures. Finally, we collected a sample of blockchain-related incidents that occurred in practice, which we further categorized using our proposed model.

Acknowledgement

This work was supported by the NRF, Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2016NCR-NCR002-028) and administered by the National Cybersecurity R&D Directorate. We would also like to thank our colleagues Pieter Hartel, Stefanos Leonardos, and Mark van Staalduinen for their valuable feedback.

References

[1] BlockVerify: Blockchain Based Anti-Counterfeit Solution. http://www.blockverify.io/, 2019.

[2] T. Abdellatif and K. Brousmiche. Formal verification of smart contracts based on users and blockchain behaviors models. In *9th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2018, Paris, France, February 26-28, 2018*, pages 1–5. IEEE, 2018.

[3] Aion Foundation. List of historical vulnerabilities.

[4] Akamai. Q1 2017 State of the Internet/Connectivity Report, 2017.

[5] A. Akentiev. Parity Multisig Hacked. Again, 2017.

[6] M. Al-Bassam and S. Meiklejohn. Contour: A practical system for binary transparency. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 94–110. Springer, 2018.

[7] E. Albert, P. Gordillo, B. Livshits, A. Rubio, and I. Sergey. Ethir: A framework for high-level analysis of Ethereum bytecode. In S. K. Lahiri and C. Wang, editors, *Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings*, volume 11138 of *Lecture Notes in Computer Science*, pages 513–520. Springer, 2018.

[8] M. Ali, J. Nelson, R. Shea, and M. J. Freedman. Bootstrapping trust in distributed systems with blockchains. *USENIX Mag.*, 41(3), 2016.

[9] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani. Applications of blockchains in the Internet of Things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 21(2):1676–1717, 2018.

[10] A. Alkhalifah, A. Ng, A. Kayes, J. Chowdhury, M. Alazab, and P. Watters. A taxonomy of blockchain threats and vulnerabilities. *Unpublished*, 2019.

[11] G. Andresen and M. Hearn. Bip-70, 2016.

[12] M. Apostolaki et al. Blockchain meets internet routing, 2017.

[13] M. Apostolaki, G. Marti, J. Müller, and L. Vanbever. SABRE: Protecting Bitcoin against routing attacks. *NDSS 2019*, 2019.

[14] M. Apostolaki, A. Zohar, and L. Vanbever. Hijacking Bitcoin: Routing attacks on cryptocurrencies. In *f*. IEEE, 2017.

[15] E. Arazi. Choosing the Right DDoS Solution (Part 4): Hybrid Protection, 2018.

[16] W. Ashford. Corporate networks vulnerable to insider attacks, report finds, 2018.

[17] N. Atzei, M. Bartoletti, and T. Cimoli. A survey of attacks on Ethereum smart contracts (SoK). In *POST*. Springer, 2017.

[18] P.-L. Aublin, S. B. Mokhtar, and V. Quéma. RBFT: Redundant byzantine fault tolerance. In *ICDCS*. IEEE, 2013.

[19] S. Avizheh, T. T. Doan, X. Liu, and R. Safavi-Naini. A secure event logging system for smart homes. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*, IoTS&#38;P '17, pages 37–42, New York, NY, USA, 2017. ACM.

[20] S. Azouvi and A. Hicks. SoK: Tools for game theoretic models of security for cryptocurrencies. *arXiv preprint arXiv:1905.08595*, 2019.

[21] S. Bag, S. Ruj, and K. Sakurai. Bitcoin block withholding attack: Analysis and mitigation. *IEEE Transactions on Information Forensics and Security*, 12(8), 2017.

[22] S. Bag and K. Sakurai. Yet another note on block withholding attack on Bitcoin mining pools. In *International Conference on Information Security*. Springer, 2016.

[23] X. Bai, Z. Cheng, Z. Duan, and K. Hu. Formal modeling and verification of smart contracts. In K. Z. Zamli, V. Mezhuyev, and L. Benedicenti, editors, *Proceedings of the 7th International Conference on Software and Computer Applications, ICSCA 2018, Kuantan, Malaysia, February 08-10, 2018*, pages 322–326. ACM, 2018.

[24] P. Bailis, A. Fekete, M. J. Franklin, A. Ghodsi, J. M. Hellerstein, and I. Stoica. Coordination avoidance in database systems. *Proceedings of the VLDB Endowment*, 8(3):185–196, 2014.

[25] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis. Consensus in the age of blockchains. *arXiv preprint arXiv:1711.03936*, 2017.

[26] J. Barry. Blockchain Technology Needs Standardization, 2018.

[27] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, pages 274–283. ACM, 2001.

[28] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing*, PODC '01, pages 274–283, New York, NY, USA, 2001. ACM.

[29] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In *CRYPTO'99*. Springer, 1999.

[30] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *USENIX Security*, 2014.

[31] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, STOC '94, pages 544–553, New York, NY, USA, 1994. ACM.

[32] J. Benet, D. Dalrymple, and N. Greco. Proof of replication. *Protocol Labs, July*, 27, 2017.

[33] I. Bentov, A. Gabizon, and A. Mizrahi. Cryptocurrencies without proof of work. In *FC*. Springer, 2016.

[34] I. Bentov, Y. Ji, F. Zhang, Y. Li, X. Zhao, L. Breidenbach, P. Daian, and A. Juels. Tesseract: Real-time cryptocurrency exchange using trusted hardware. *IACR Cryptology ePrint Archive*, 2017:1153, 2017.

[35] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending Bitcoin's proof of work via proof of stake. In *ACM SIGMETRICS*, 2014.

[36] I. Bentov, R. Pass, and E. Shi. Snow white: Provably secure proofs of stake., 2016.

[37] H. Berg, T. A. Proebsting, et al. Hanson's automated market maker. *Journal of Prediction Markets*, 3(1):45–59, 2009.

[38] A. Bessani, J. Sousa, and E. E. Alchieri. State machine replication for the masses with bft-smart. In *IEEE/IFIP DSN*. IEEE, 2014.

[39] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, N. Kulatova, A. Rastogi, T. Sibut-Pinote, N. Swamy, and S. Z. Béguelin. Formal verification of smart contracts: Short paper. In T. C. Murray and D. Stefan, editors, *Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, PLAS@CCS 2016, Vienna, Austria, October 24, 2016*, pages 91–96. ACM, 2016.

[40] D. Bhowmik and T. Feng. The multimedia blockchain: A distributed and tamper-proof media transaction framework. In *2017 22nd International Conference on Digital Signal Processing (DSP)*, pages 1–5, Aug 2017.

[41] A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of clients in Bitcoin P2P network. In *ACM CCS*. ACM, 2014.

[42] A. Biryukov and I. Pustogarov. Bitcoin over Tor isn't a good idea. *CoRR*, abs/1410.6079, 2014.

[43] G. Bissias, A. P. Ozisik, B. N. Levine, and M. Liberatore. Sybil-resistant mixing for Bitcoin. In *WPES*, New York, NY, USA, 2014.

[44] Bitcoin Exchange Guide News Team. Litecoin cash (lcc) experiences 51% attack, cryptocurrency dies on spot, 2018.

[45] Bitcoin Project. Bitcoin Core, 2018.

[46] Bitcoin Wiki. Value overflow incident, 2016.

[47] Bitcoin Wiki. Atomic Swap, 2018.

[48] Bitcoin.com. Bitcoin.com notary, 2019.

[49] Bitcoinj team. Bitcoinj security model, 2019.

[50] BitcoinWiki. Peercoin, 2019.

[51] BitcoinWiki. Weaknesses, 24 July 2018.

[52] BitLox. BitLox wallet, 2018.

[53] E.-O. Blass and F. Kerschbaum. Strain: A secure auction for blockchains. In *European Symposium on Research in Computer Security*, pages 87–110. Springer, 2018.

[54] Blockchain Luxembourg S.A. Blockchain Wallet, 2018.

[55] Blockusign Team. Blockusign: Encrypted document signing and digital notary – powered by the blockchain, 2019.

[56] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *ASIACRYPT*. Springer, 2001.

[57] J. Bonneau, E. W. Felten, S. Goldfeder, J. A. Kroll, and A. Narayanan. Why buy when you can rent? Bribery attacks on Bitcoin consensus. In *Proceedings of the 3rd Workshop on Bitcoin and Blockchain Research (BITCOIN '16)*, 2016.

[58] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. Sok: Research perspectives and challenges for Bitcoin and cryptocurrencies. In *IEEE S&P*. IEEE, 2015.

[59] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten. Mixcoin: Anonymity for Bitcoin with accountable mixes. In *FC*, pages 486–504. Springer, 2014.

[60] P. Boucher. What if blockchain technology revolutionised voting. *Unpublished manuscript, European Parliament*, 2016.

[61] A. Boverman. Timejacking & Bitcoin, 2011.

[62] S. Box and J. K. West. Economic and social benefits of internet openness, 22 June 2016.

[63] D. Bradbury. Hackers hit Bitcoin central exchange, 2013.

[64] S. Bradshaw and L. DeNardis. The politicization of the Internet's Domain Name System: Implications for Internet security, universality, and freedom , 2016.

[65] D. Breitenbacher, I. Homoliak, Y. L. Aung, N. O. Tippenhauer, and Y. Elovici. HADES-IoT: A Practical Host-Based Anomaly Detection System for IoT Devices. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019*, pages 479–484, 2019.

[66] L. Brent, A. Jurisevic, M. Kong, E. Liu, F. Gauthier, V. Gramoli, R. Holz, and B. Scholz. Vandal: A scalable security analysis framework for smart contracts. *CoRR*, abs/1809.03981, 2018.

[67] J. Brown-Cohen, A. Narayanan, A. Psomas, and S. M. Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 459–473. ACM, 2019.

[68] BTC.com. BTC wallet: Powerful Bitcoin and Bitcoin Cash wallet, 2018.

[69] E. Buchman, J. Kwon, and Z. Milosevic. The latest gossip on BFT consensus, 2018.

[70] P. Buneman, S. Khanna, and T. Wang-Chiew. Why and where: A characterization of data provenance. In *International conference on database theory*, pages 316–330. Springer, 2001.

[71] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh. Zether: Towards privacy in a smart contract world. *IACR Cryptology ePrint Archive*, 2019:191, 2019.

[72] S. C. Bürgel. Rock paper scissors on Ethereum, 2016.

[73] V. Buterin. Long-range attacks: The serious problem with adaptive proof of work. *Ethereum Blog, May*, 2014.

[74] V. Buterin. Vyper. https://vyper.readthedocs.io/en/v0.1.0-beta.9/, 2017.

[75] V. Buterin and V. Griffith. Casper the friendly finality gadget, 2017.

[76] A. Butler. Ethereum Classic attacked! how does the 51% attack occur?, 2019.

[77] C. Cachin. Yet another visit to paxos. *IBM Research, Zurich, Switzerland, Tech. Rep. RZ3754*, 2009.

[78] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. *Journal of Cryptology*, 18(3), 2005.

[79] C. Cachin and J. A. Poritz. Secure intrusion-tolerant replication on the internet. In *DSN*. IEEE, 2002.

[80] C. Cachin and M. Vukolić. Blockchain consensus protocols in the wild, 2017.

[81] G. Caffyn. Chainalysis CEO Denies Sybil Attack on Bitcoin Network, 2015.

[82] D. Canellis. Bitcoin wallet Electrum hit by DoS attack from 140,000-strong botnet, 2019.

[83] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In B. S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, pages 90–104, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[84] R. Canetti and R. Gennaro. Incoercible multiparty computation. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 504–513, Oct 1996.

[85] D. Carboni. Feedback based reputation on top of the Bitcoin blockchain. *CoRR*, abs/1502.01504, 2015.

[86] CarbonWallet.com. Multi Signature Online Cryptocurrency Wallet, 2018.

[87] M. Carlsten, H. A. Kalodner, S. M. Weinberg, and A. Narayanan. On the instability of Bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[88] F. Casino, T. K. Dasaklis, and C. Patsakis. A systematic literature review of blockchain-based applications: current status, classification and open issues. *Telematics and Informatics*, 2018.

[89] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, 1999.

[90] R. Cavanagh. Federal Information Processing Standard (FIPS) 186-4, Digital Signature Standard; Request for Comments on the NIST-Recommended Elliptic Curves. Technical report, National Institute of Standards and Technology, 2015.

[91] J. Chang, B. Gao, H. Xiao, J. Sun, Y. Cai, and Z. Yang. sCompile: Critical path identification and analysis for smart contracts. In *Formal Methods and Software Engineering - 21th International Conference on Formal Engineering Methods, ICFEM 2019, Shenzhen, China, November 5th-9th, 2019, Proceedings*. Springer, 2019.

[92] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988.

[93] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, Feb. 1981.

[94] H. Chen, M. Pendleton, L. Njilla, and S. Xu. A survey on Ethereum systems security: Vulnerabilities, attacks and defenses. *arXiv preprint arXiv:1908.04507*, 2019.

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

[96] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution. *arXiv preprint arXiv:1804.05141*, 2018.

[97] U. Chohan. The concept and criticisms of steemit, 2018.

[98] N. Christin. Traveling the silk road: A measurement analysis of a large anonymous online marketplace. In *Proceedings of the 22nd international conference on World Wide Web*, pages 213–224. ACM, 2013.

[99] Circle Internet Financial Limited. Circle Pay, 2018.

[100] Citowise Developments. Citowise wallet, 2018.

[101] J. Clark and A. Essex. Commitcoin: Carbon dating commitments with Bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 390–398. Springer, 2012.

[102] J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, SFCS '85, pages 372–382, Washington, DC, USA, 1985. IEEE Computer Society.

[103] Coinbase. Coinbase wallet, 2018.

[104] M. L. Collins, M. C. Theis, R. F. Trzeciak, J. R. Strozer, J. W. Clark, D. L. Costa, T. Cassidy, M. J. Albrethsen, and A. P. Moore. Common sense guide to prevention and detection of insider threats 5th edition. *Published by CERT, SEI, CMU*, 2016.

[105] Common Criteria. Common criteria for information technologysecurity evaluation. part 1: Introduction and general model, 2017.

[106] Consensys. Security tools, 2016.

[107] Consensys team. Ethereum MultiSigWallet, 2017.

[108] M. Conti, E. S. Kumar, C. Lal, and S. Ruj. A survey on security and privacy issues of Bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4), 2018.

[109] CoolBitX. The CoolWallet S, 2018.

[110] K. Crispin. Alt-roots, alt-tlds. IETF Draft, 2001.

[111] S. A. Crosby and D. S. Wallach. Efficient data structures for tamper-evident logging. In *USENIX Security Symposium*, pages 317–334, 2009.

[112] crytic. Slither, the Solidity source analyzer, 2018.

[113] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash Boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. *arXiv preprint arXiv:1904.05234*, 2019.

[114] P. Daian, R. Pass, and E. Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proofs of stake, 2017.

[115] B. David, P. Gaži, A. Kiayias, and A. Russell. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT*. Springer, 2018.

[116] A. S. de Pedro, D. Levi, and L. I. Cuende. Witnet: A decentralized oracle network protocol, 2017.

[117] P. Delaney. Major attack reported on dogecoin vault leads to shutdown, 2014.

[118] Dell SecureWorks. Cryptocurrency-stealing malware landscape, 2015.

[119] Deloitte. Continuous interconnected supply chain Using Blockchain and Internet-of-Things in supply chain traceability, 2017.

[120] M. Di Angelo and G. Salzer. A survey of tools for analyzing Ethereum smart contracts. In *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*. IEEE, 2019.

[121] A. Dorri, S. S. Kanhere, and R. Jurdak. Towards an optimized blockchain for IoT. In *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, pages 173–178. ACM, 2017.

[122] J. R. Douceur. The sybil attack. In *IPTPS*. Springer, 2002.

[123] R. Dua. Solium documentation release 1.0.0, 11 Feb 2019.

[124] S. Duan, H. Meling, S. Peisert, and H. Zhang. Bchain: Byzantine replication with high throughput and embedded reconfiguration. In *OPODIS*. Springer, 2014.

[125] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak. Proofs of space. In *CRYPTO'15*, 2015.

[126] K. Elby. King of the Ether Throne: Post mortem investigation, 2016.

[127] Electrum Technologies GmbH. Electrum Bitcoin wallet, 2018.

[128] ELLIPAL. ELLIPAL Hardware Wallet 2.0, 2019.

[129] S. Ellis, A. Juels, and S. Nazarov. ChainLink: A Decentralized Oracle Network, 2017.

[130] Enterprise Ethereum Alliance. Enterprise Ethereum Alliance, 2019.

[131] ERC-725 Alliance. Erc-725: Ethereum identity standard, 2018.

[132] S. Eskandari, J. Clark, D. Barrera, and E. Stobert. A first look at the usability of Bitcoin key management, 2018.

[133] T. Espel, L. Katz, and G. Robin. Proposal for protocol on a quorum blockchain with zero knowledge., 2017.

[134] C. Esposito, A. De Santis, G. Tortora, H. Chang, and K.-K. R. Choo. Blockchain: A panacea for healthcare cloud-based data security and privacy? *IEEE Cloud Computing*, 5(1):31–37, 2018.

[135] EtherDelta team. EtherDelta, 2019.

[136] Ethereum Foundation. Lisp like language, 2019.

[137] Ethereum Foundation. Yul, 2019.

[138] Ethereum team. Solidity.

[139] Ethereum team. A Next-Generation Smart Contract and Decentralized Application Platform. https://github.com/ethereum/wiki/wiki/White-Paper#modified-ghost-implementation, 2018.

[140] Ethersphere. Swarm documentation, 2018.

[141] Etherum team. Serpent, 2017.

[142] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7), 2018.

[143] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar. A survey on privacy protection in blockchain system. *Journal of Network and Computer Applications*, 126, 2019.

[144] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke. Blockchain Technologies for the Internet of Things: Research Issues and Challenges. *IEEE Internet of Things Journal*, 6(2):2188–2204, April 2019.

[145] D. Fisher. Final report on diginotar hack shows total compromise of CA servers. *Retrieved September*, 8:2013, 2012.

[146] D. Floyd. 150K Stolen From MyEtherWallet Users in DNS Server Hijacking, 2018.

[147] M. Franklin. A survey of key evolving cryptosystems. *International Journal of Security and Networks*, 1(1-2), 2006.

[148] C. Fromknecht, D. Velicanu, and S. Yakoubov. A decentralized public key infrastructure with identity retention. *IACR Cryptology ePrint Archive*, 2014:803, 2014.

[149] E. Gaetani, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone. Blockchain-based database to ensure data integrity in cloud computing environments. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), Venice, Italy, January 17-20, 2017.*, pages 146–155, 2017.

[150] H. S. Galal and A. M. Youssef. Succinctly verifiable sealed-bid auction smart contract. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 3–19. Springer, 2018.

[151] H. S. Galal and A. M. Youssef. Verifiable sealed-bid auction on the Ethereum blockchain. In *International Conference on Financial Cryptography and Data Security*, pages 265–278. Springer, 2018.

[152] H. S. Galal and A. M. Youssef. Trustee: full privacy preserving vickrey auction on top of Ethereum. *arXiv preprint arXiv:1905.06280*, 2019.

[153] J. A. Garay and A. Kiayias. SoK: A consensus taxonomy in the blockchain era. *IACR Cryptology ePrint Archive*, 2018:754, 2018.

[154] P. Gaži, A. Kiayias, and A. Russell. Stake-bleeding attacks on proof-of-stake blockchains. In *IEEE CVCBT*. IEEE, 2018.

[155] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *SOSP*. ACM, 2017.

[156] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News*, 33(2):51–59, 2002.

[157] B. Gipp, N. Meuschke, and A. Gernandt. Decentralized trusted timestamping using the crypto currency Bitcoin. *CoRR*, abs/1502.04015, 2015.

[158] D. Giry. Cryptographic Key Length Recommendations, 2019.

[159] Gnosis Team. Gnosis-whitepaper. *URL: https://gnosis. pm/resources/default/pdf/gnosis_whitepaper. pdf*, 2017.

[160] L. Goasduff. Gartner Says Blockchain Deployments Across Financial Services Ecosystems Are At Least Three Years Away, 2019.

[161] S. Goldfeder, J. Bonneau, R. Gennaro, and A. Narayanan. Escrow protocols for cryptocurrencies: How to buy physical goods using Bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 321–339. Springer, 2017.

[162] S. Goldfeder, R. Gennaro, H. Kalodner, J. Bonneau, J. A. Kroll, E. W. Felten, and A. Narayanan. Securing Bitcoin wallets via a new DSA/ECDSA threshold signature scheme, 2015.

[163] S. Goldlust et al. Bind 9 security vulnerability matrix, 2019.

[164] A. Grech and A. F. Camilleri. Blockchain in education, 2017.

[165] N. Grech, M. Kong, A. Jurisevic, L. Brent, B. Scholz, and Y. Smaragdakis. Madmax: surviving out-of-gas conditions in Ethereum smart contracts. *PACMPL*, 2(OOPSLA):116:1–116:27, 2018.

[166] A. Greenberg. Hacker redirects traffic from 19 internet providers to steal Bitcoins, 2014.

[167] I. Grishchenko, M. Maffei, and C. Schneidewind. A semantic framework for the security analysis of Ethereum smart contracts. In L. Bauer and R. Küsters, editors, *Principles of Security and Trust - 7th International Conference, POST 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, volume 10804 of *Lecture Notes in Computer Science*, pages 243–269. Springer, 2018.

[168] J. Groth. Efficient maximal privacy in boardroom voting and anonymous broadcast. In A. Juels, editor, *Financial Cryptography*, pages 90–104, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[169] GuardRails. Write to arbitrary storage locations.

[170] J. Guarnizo and P. Szalachowski. PDFS: practical data feed service for smart contracts. In *Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part I*, pages 767–789, 2019.

[171] D. Gutteridge. Japanese cryptocurrency monacoin hit by selfish mining attack, 2018.

[172] T. Hanke, M. Movahedi, and D. Williams. Dfinity technology overview series, consensus system, 2018.

[173] T. Hardjono and N. Smith. Cloud-based commissioning of constrained devices using permissioned blockchains. In *Proceedings of the 2nd ACM international workshop on IoT privacy, trust, and security*, pages 29–36. ACM, 2016.

[174] P. Hartel and M. van Staalduinen. Truffle tests for free – replaying Ethereum smart contracts for transparency. Technical report, Singapore Univ. of Technology and Design, Singapore, Jul 2019.

[175] D. Harz and W. Knottenbelt. Towards safer smart contracts: A survey of languages and verification methods. *arXiv preprint arXiv:1809.09805*, 2018.

[176] E. Heilman, F. Baldimtsi, and S. Goldberg. Blindly signed contracts: Anonymous on-blockchain and off-blockchain Bitcoin transactions. In *FC*. Springer, 2016.

[177] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse attacks on Bitcoin's peer-to-peer network. In *USENIX Security*, pages 129–144, 2015.

[178] Heleness. Blockchain-based notarization platform on Ethereum, 2018.

[179] L. Hellebrandt, I. Homoliak, K. Malinka, and P. Hanáček. Increasing trust in Tor node list using blockchain. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 29–32. IEEE, 2019.

[180] M. Herlihy. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 245–254. ACM, 2018.

[181] M. P. Herlihy and J. M. Wing. Axioms for concurrent objects. In *ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. ACM, 1987.

[182] A. Hertig. 'Bitcoin bug' exploited on crypto fork as attacker prints 235 million pigeoncoins, 2018.

[183] E. Hertzog, G. Benartzi, and G. Benartzi. Bancor Protocol: A Hierarchical Monetary System and the Foundation of a Global Decentralized Autonomous Exchange, 2017.

[184] S. Higgins. Bitcoin mining pools targeted in wave of DDoS attacks, 2015.

[185] S. Higgins. The bitfinex Bitcoin hack: What we know (and don't know), 2016.

[186] F. Hildebrand. The state of ponzi schemes, 2018.

[187] E. Hildenbrandt, M. Saxena, X. Zhu, N. Rodrigues, P. Daian, D. Guth, and G. Rosu. Kevm: A complete semantics of the Ethereum virtual machine, 2017.

[188] I. Homoliak, D. Breitenbacher, A. Binder, and P. Szalachowski. An air-gapped 2-factor authentication for smart-contract wallets, 2018.

[189] I. Homoliak, S. Venugopalan, Q. Hum, and P. Szalachowski. A security reference architecture for blockchains. In *The 2nd IEEE International Conference on Blockchain (Blockchain-2019)*. IEEE, 2019.

[190] T. Hønsi. Spacemint-a cryptocurrency based on proofs of space. Master's thesis, NTNU, 2017.

[191] J. Huang, K. Lei, M. Du, H. Zhao, H. Liu, J. Liu, and Z. Qi. Survey on blockchain incentive mechanism. In *International Conference of Pioneering Computer Scientists, Engineers and Educators*, pages 386–395. Springer, 2019.

[192] Hyperledger team. Hyperledger architecture, volume 1: Consensus, 2017.

[193] S. Inc. Shocard: Identity management verified using the blockchain, 2017.

[194] ISO. ISO/CD 23257: Blockchain and distributed ledger technologies – Reference architecturevulnerabilities, 2019.

[195] ISO. ISO/DTR 23245: Blockchain and distributed ledger technologies – Security risks, threats and vulnerabilities, 2019.

[196] C. Jaag and C. Bach. Blockchain technology and cryptocurrencies: Opportunities for postal financial services. In *The Changing Postal and Delivery Sector*, pages 205–221. Springer, 2017.

[197] C. Jentzsch. The history of the DAO and lessons learned, 2016.

[198] B. Jiang, Y. Liu, and W. Chan. Contractfuzzer: Fuzzing smart contracts for vulnerability detection. In *ASE*. ACM, 2018.

[199] N. Johnson. Euler: The simplest exchange and token currency, 2016.

[200] A. Judmayer, N. Stifter, A. Zamyatin, I. Tsabary, I. Eyal, P. Gaži, S. Meiklejohn, and E. Weippl. Pay-to-win: Incentive attacks on proof-of-work cryptocurrencies. *IACR Cryptology ePrint Archive*, 2019.

[201] O. R. Kabi and V. N. Franqueira. Blockchain-based distributed marketplace. In *International Conference on Business Information Systems*, pages 197–210. Springer, 2018.

[202] I. Kadyrov. How to test Ethereum smart contracts, 2019.

[203] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan. An empirical study of Namecoin and lessons for decentralized namespace design. In *WEIS*. Citeseer, 2015.

[204] S. Kalra, S. Goel, M. Dhawan, and S. Sharma. ZEUS: analyzing safety of smart contracts. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[205] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn. An empirical analysis of anonymity in zcash. In *USENIX Security*, 2018.

[206] K. Karantias, A. Kiayias, and D. Zindros. Proof-of-burn. *IACR Cryptology ePrint Archive*, 2019. https://eprint.iacr.org/2019/1096.pdf.

[207] KeepKey. The Simple Cryptocurrency Hardware Wallet, 2018.

[208] D. Khader, B. Smyth, P. Y. A. Ryan, and F. Hao. A fair and robust voting system by broadcast. In *5th International Conference on Electronic Voting 2012, (EVOTE 2012), Co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC, July 11-14, 2012, Castle Hofen, Bregenz, Austria*, pages 285–299, 2012.

[209] S. Khandelwal. Danish Bitcoin exchange bips hacked and 1,295 Bitcoins worth $1 million stolen, 2013.

[210] A. Kiayias and A. Russell. Ouroboros-BFT: A Simple Byzantine Fault Tolerant Consensus Protocol, 2018.

[211] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO'17*. Springer, 2017.

[212] A. Kiayias and M. Yung. Self-tallying elections and perfect ballot secrecy. In D. Naccache and P. Paillier, editors, *Public Key Cryptography*, pages 141–158, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[213] H. M. Kim and M. Laskowski. Towards an ontology-driven blockchain design for supply chain provenance. *CoRR*, abs/1610.02922, 2016.

[214] J. C. King. Symbolic execution and program testing. *Commun. ACM*, 19(7):385–394, 1976.

[215] R. Klomp and A. Bracciali. On symbolic verification of Bitcoin's script language. In J. García-Alfaro, J. Herrera-Joancomartí, G. Livraga, and R. Rios, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings*, volume 11025 of *Lecture Notes in Computer Science*, pages 38–56. Springer, 2018.

[216] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing Bitcoin security and performance with strong consistency via collective signing. In *USENIX Security*, 2016.

[217] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE S&P 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, 2018.

[218] T. Kolinko. Eveem, 2019.

[219] Komodo team. Komodo, 2019.

[220] H. Kopp, C. Bösch, and F. Kargl. Koppercoin–a distributed file storage with financial incentives. In *International Conference on Information Security Practice and Experience*, pages 79–93. Springer, 2016.

[221] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *IEEE S&P*. IEEE, 2016.

[222] J. Krupp and C. Rossow. teether: Gnawing at Ethereum to automatically exploit smart contracts. In *USENIX Security*, 2018.

[223] N. Kshetri. 1 blockchain's roles in meeting key supply chain management objectives. *International Journal of Information Management*, 39:80 – 89, 2018.

[224] P. Labs. Interplanetary file system, 2018.

[225] L. Lamport et al. The part-time parliament. *ACM Transactions on Computer systems*, 16(2), 1998.

[226] B. Laurie, A. Langley, and E. Kasper. Certificate transparency. *ACM Queue*, 12(8):10–19, 2014.

[227] Ledger. Ledger Nano S, 2018.

[228] S. Leonardos, D. Reijsbergen, and G. Piliouras. PREStO: A systematic framework for blockchain consensus protocols. *arXiv preprint arXiv:1906.06540*, 2019.

[229] M. Lepinski and K. Sriram. Bgpsec protocol specification. RFC 8205, RFC Editor, 2017.

[230] S. D. Lerner. New DoS vuln by Forcing Continuous Hard Disk Seek/Read Activity (fixed in 0.8.0), 2019.

[231] W. Li, S. Andreina, J.-M. Bohli, and G. Karame. Securing proof-of-stake blockchain protocols. In *DPM*. Springer, 2017.

[232] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, 2017.

[233] Y. Li, W. Susilo, G. Yang, Y. Yu, D. Liu, and M. Guizani. A blockchain-based self-tallying voting scheme in decentralized IoT. *CoRR*, abs/1902.03710, 2019.

[234] G. Liang, S. R. Weller, F. Luo, J. Zhao, and Z. Y. Dong. Distributed blockchain-based data protection framework for modern power systems against cyber attacks. *IEEE Transactions on Smart Grid*, 10(3):3162–3173, 2018.

[235] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu. Blockchain based data integrity service framework for IoT data. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 468–475, June 2017.

[236] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf. NIST cloud computing reference architecture. *NIST special publication*, 500(2011):1–28, 2011.

[237] C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, and M. Sena. uPort: a platform for self-sovereign identity, 2016.

[238] Luno. Luno wallet, 2018.

[239] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making smart contracts smarter. In *ACM CCS*. ACM, 2016.

[240] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30. ACM, 2016.

[241] J. Ma, B. Qi, and K. Lv. Fully private auctions for the highest bid. In *Proceedings of the ACM Turing Celebration Conference-China*, page 64. ACM, 2019.

[242] Mahendar B. Unexpected ether, 2019.

[243] A. Manning. Solidity security: Comprehensive list of known attack vectors and common anti-patterns. https://blog.sigmaprime.io/solidity-security.html, 2018.

[244] Y. Marcus, E. Heilman, and S. Goldberg. Low-resource eclipse attacks on Ethereum's peer-to-peer network., 2018.

[245] A. Mavridou and A. Laszka. Designing secure Ethereum smart contracts: A finite state machine based approach. *CoRR*, abs/1711.09327, 2017.

[246] G. Maxwell. Coinjoin: Bitcoin privacy for the real world. In *Post on Bitcoin forum*, 2013.

[247] D. Maziéres. The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus, 2015.

[248] P. McCorry, S. F. Shahandashti, and F. Hao. Refund attacks on Bitcoin's payment protocol. In *International Conference on Financial Cryptography and Data Security*, pages 581–599. Springer, 2016.

[249] P. McCorry, S. F. Shahandashti, and F. Hao. A smart contract for boardroom voting with maximum voter privacy. In *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*, pages 357–375, 2017.

[250] L. W. McKnight and J. P. Bailey. An introduction to internet economics. In *The Journal of Electronic Publishing*. Michigan Publishing, 1995.

[251] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 120–130. IEEE, 1999.

[252] A. Miller. Feather-forks: enforcing a blacklist with sub - 50% hash power. https://bitcointalk.org/index.php?topic=312668.0, 2013.

[253] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz. Permacoin: Repurposing Bitcoin work for data preservation. In *IEEE S&P*. IEEE, 2014.

[254] A. Miller, A. Kosba, J. Katz, and E. Shi. Nonoutsourceable scratch-off puzzles to discourage Bitcoin mining coalitions. In *ACM CCS*. ACM, 2015.

[255] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee. Discovering Bitcoin' s public topology and influential nodes. Technical report, University of Maryland, College Park, 2015.

[256] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song. The honey badger of bft protocols. In *ACM CCS*. ACM, 2016.

[257] G. Milton. Hackers launch 'double-spend' attack on Bitcoin gold to steal over $18 million, 2018.

[258] A. Mizrahi. A blockchain-based property ownership recording system. *A Blockchain-based Property Ownership Recording System*, 2015.

[259] MME. Conceptual framework for legal and risk assessment of crypto tokens, 2018.

[260] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, K. Hogan, J. Hennessey, A. Miller, A. Narayanan, et al. An empirical analysis of traceability in the Monero blockchain. *PETS*, 2018(3), 2018.

[261] B. Mueller. Mythril classic, 2017.

[262] Y. Murray and D. A. Anisi. Survey of formal verification methods for smart contracts on blockchain. In *10th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2019, Canary Islands, Spain, June 24-26, 2019*, pages 1–6. IEEE, 2019.

[263] Mycelium Holding LTD. Mycelium wallet, 2018.

[264] MyEtherWallet, Inc. MyEtherWallet, 2018.

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

[266] Namecoin Team. Namecoin: Decentralize all the things!, 2019.

[267] A. Narayanan. Analyzing the 2013 Bitcoin fork: centralized decision-making saved the day, 2018.

[268] National Notary Association, CA. What is notarization?, 2019.

[269] C. Natoli, J. Yu, V. Gramoli, and P. Esteves-Verissimo. Deconstructing blockchains: A comprehensive survey on consensus, membership and structure. *arXiv preprint arXiv:1908.08316*, 2019.

[270] K. Nayak, S. Kumar, A. Miller, and E. Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *IEEE EuroSP*. IEEE, 2016.

[271] Z. Nehai, P. Piriou, and F. F. Daumas. Model-checking of smart contracts. In *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), iThings/GreenCom/CPSCom/SmartData 2018, Halifax, NS, Canada, July 30 - August 3, 2018*, pages 980–987. IEEE, 2018.

[272] T. Neudecker and H. Hartenstein. Network layer aspects of permissionless blockchains. *IEEE Communications Surveys & Tutorials*, 21(1):838–857, 2018.

[273] I. Nikolić, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor. Finding the greedy, prodigal, and suicidal contracts at scale. In *ACM ACSAC*. ACM, 2018.

[274] S. Noether. Ring signature confidential transactions for monero, 2015.

[275] T. Nolan. Alt chains and atomic transfers, 2013.

[276] B. Notheisen, J. B. Cholewa, and A. P. Shanmugam. Trading real-world assets on blockchain. *Business & Information Systems Engineering*, 59(6):425–440, 2017.

[277] o0ragman0o. ITT – Intrinsically Tradable Token, 2017.

[278] ocminer. Network Attack on XVG / VERGE, 2018.

[279] R. O'Connor. Simplicity: A new language for blockchains. In *Proceedings of the 2017 Workshop on Programming Languages and Analysis for Security, PLAS@CCS 2017, Dallas, TX, USA, October 30, 2017*, pages 107–120. ACM, 2017.

[280] T. of Bits. Manticore documentation release 0.1.0, 2019.

[281] H. Official. Unit underflows and overflows – Ethereum solidity vulnerability, 2018.

[282] B. M. Oki and B. H. Liskov. Viewstamped replication: A new primary copy method to support highly-available distributed systems. In *ACM Symposium on Principles of Distributed Computing*. ACM, 1988.

[283] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. In *USENIX ATC*, 2014.

[284] OpenBazaar Team. Escrow Smart Contract Specification in OpenBazaar, 2018.

[285] OpenTimestamps Team. OpenTimestamps. https://opentimestamps.org/, 2019.

[286] P2Pool.org. P2Pool – Decentralized Bitcoin Mining Pool, 2017.

[287] P4Titan. Slimcoin: A peer-to-peer crypto-currency with proof-of-burn. Technical report, 2014.

[288] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT'99*, Berlin, Heidelberg, 1999.

[289] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito. Blockchain and IoT integration: A systematic survey. *Sensors*, 18(8):2575, 2018.

[290] R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, and A. Singh. Empirical vulnerability analysis of automated smart contracts security testing on blockchains. In *CASCON*. IBM Corp., 2018.

[291] S. Park, K. Pietrzak, J. Alwen, G. Fuchsbauer, and P. Gazi. Spacecoin: A cryptocurrency based on proofs of space. Technical report, 2015.

[292] R. Pass and E. Shi. Fruitchains: A fair blockchain. In *PODC*. ACM, 2017.

[293] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *CRYPTO'91*, Berlin, Heidelberg, 1992.

[294] J. Peterson and J. Krug. Augur: a decentralized, open-source platform for prediction markets. *arXiv preprint arXiv:1501.01042*, 2015.

[295] A. Peyton. Cyren sounds siren over Bitcoin siphon scam, 2017.

[296] POEX.IO Team. Proof of existence, 2019.

[297] L. Poinsignon. BGP leaks and cryptocurrencies, 2018.

[298] J. Poon and T. Dryja. The Bitcoin lightning network: Scalable off-chain instant payments, 2016.

[299] S. Popejoy. The Pact smart contract language. http://kadena.io/docs/Kadena-PactWhitepaper.pdf, 2016.

[300] S. Popov. The Tangle. http://tanglereport.com/wp-content/uploads/2018/01/IOTA_Whitepaper.pdf, 2016.

[301] P. Pritzker and W. E. May. Secure Hash Standard (SHS). Technical report, National Institute of Standards and Technology, 2015.

[302] Proovable Team. The Provable blockchain oracle for modern DApps, 2019.

[303] Protocol Labs. Filecoin: A decentralized storage network. Technical report, Protocol Labs, 2017.

[304] Protofire. Solhint project, 2017.

[305] I. Pustogarov. *Deanonymisation techniques for Tor and Bitcoin*. PhD thesis, University of Luxembourg, 2015.

[306] QuantumMechanic. Proof of stake instead of proof of work, 2011.

[307] Rachel Abrams and Nathaniel Popper. Trading Site Failure Stirs Ire and Hope for Bitcoin, 2014.

[308] K. Rantos, G. Drosatos, K. Demertzis, C. Ilioudis, and A. Papanikolaou. Blockchain-based consents management for personal data processing in the IoT ecosystem. In *ICETE (2)*, pages 738–743, 2018.

[309] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. *A scalable content-addressable network*, volume 31. ACM, 2001.

[310] Raulo. Optimal pool abuse strategy. http://bitcoin.atspace.com/poolcheating.pdf, 2011.

[311] F. Restuccia, S. D. Kanhere, T. Melodia, and S. K. Das. Blockchain for the Internet of Things: Present and future. *arXiv preprint arXiv:1903.07448*, 2019.

[312] Reuters. Bitcoin Worth $72M was Stolen in Bitfinex Exchange Hack in Hong Kong, 2016.

[313] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2001.

[314] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In C. Boyd, editor, *ASIACRYPT*, Berlin, Heidelberg, 2001.

[315] P. R. Rizun. Subchains: A technique to scale Bitcoin and improve the user experience. *Ledger*, 1, 2016.

[316] R. Rodrigues and P. Druschel. Peer-to-peer systems. *Commun. ACM*, 53(10), 2010.

[317] M. Rosenfeld. Analysis of Bitcoin pooled mining reward systems. *CoRR*, abs/1112.4980, 2011.

[318] D. S. H. Rosenthal, P. Maniatis, M. Roussopoulos, T. J. Giuli, and M. Baker. Notes on the design of an internet adversary. *CoRR*, cs.DL/0411078, 2004.

[319] T. Ruffing, P. Moreno-Sanchez, and A. Kate. Coinshuffle: Practical decentralized coin mixing for Bitcoin. In *ESORICS*. Springer, 2014.

[320] S. King and S. Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. Technical report, 2012.

[321] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang, and A. Mohaisen. Exploring the attack surface of blockchain: A systematic overview. *arXiv preprint arXiv:1904.03487*, 2019.

[322] K. Sako and J. Kilian. Receipt-free mix-type voting scheme. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology — EUROCRYPT '95*, pages 393–403, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[323] A. Sapirshtein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in Bitcoin. In *FC*. Springer, 2016.

[324] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *IEEE S&P*. IEEE, 2014.

[325] A. Schaub, R. Bazin, O. Hasan, and L. Brunie. A trustless privacy-preserving reputation system. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 398–411. Springer, 2016.

[326] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM CSUR*, 22(4), 1990.

[327] F. Schrans, S. Eisenbach, and S. Drossopoulou. Writing safe smart contracts in Flint. In S. Marr and J. B. Sartor, editors, *Conference Companion of the 2nd International Conference on Art, Science, and Engineering of Programming, Nice, France, April 09-12, 2018*, pages 218–219. ACM, 2018.

[328] O. Schrijvers, J. Bonneau, D. Boneh, and T. Roughgarden. Incentive compatibility of Bitcoin mining pool reward functions. In *Financial Crypto*. Springer, 2016.

[329] D. Schwartz, N. Youngs, A. Britto, et al. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, 5, 2014.

[330] K. Sedgwick. Verge is forced to fork after suffering a 51% attack, 2018.

[331] I. Sergey, A. Kumar, and A. Hobor. Scilla: a smart contract intermediate-level language. *arXiv preprint arXiv:1801.00687*, 2018.

[332] A. Shahaab, B. Lidgey, C. Hewage, and I. Khan. Applicability and appropriateness of distributed ledgers consensus protocols in public and private sectors: A systematic review. *IEEE Access*, 7:43622–43636, 2019.

[333] D. Shares. Gatecoin official statement: Hot wallet breach losses estimated to be $2m usd, 2016.

[334] D. Shares. Major DDoS attacks hit Bitcoin.com, 2017.

[335] P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park. Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks. *IEEE Communications Magazine*, 55(9):78–85, 2017.

[336] S. Shetty, V. Red, C. Kamhoua, K. Kwiat, and L. Njilla. Data provenance assurance in the cloud using blockchain. In *Disruptive Technologies in Sensors and Sensor Systems*, volume 10206, page 102060I. International Society for Optics and Photonics, 2017.

[337] V. Shoup. Practical threshold signatures. In *EUROCRYPT*. Springer, 2000.

[338] SIGANY LTD. OpenTimestamps. https://silentnotary.com/, 2019.

[339] SmartContractSecurity. Smart contract weakness classification registry, 2019.

[340] J. Smith. Echidna, 2018.

[341] Y. Sompolinsky, Y. Lewenberg, and A. Zohar. Spectre: Serialization of proof-of-work events: confirming transactions via recursive elections, 2016.

[342] Y. Sompolinsky and A. Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. *IACR Cryptology ePrint Archive*, 2013(881), 2013.

[343] S. Son and V. Shmatikov. The Hitchhiker's Guide to DNS Cache Poisoning. In S. Jajodia and J. Zhou, editors, *SecureComm*, Berlin, Heidelberg, 2010.

[344] STAMPD Team. Stampd.io, 2019.

[345] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.

[346] R. Stortz. Rattle: a binary static analysis framework, 2018.

[347] M. Suiche. Porosity: A decompiler for blockchain-based smart contracts bytecode. *DEF CON*, 25, 2017.

[348] M. Sutton, A. Greene, and P. Amini. *Fuzzing: brute force vulnerability discovery*. Pearson Education, 2007.

[349] D. Swift. A Practical Application ofSIM/SEM/SIEM Automating Threat Identification, 2006).

[350] N. Szabo. The idea of smart contracts, 1997.

[351] P. Szalachowski. Blockchain-based TLS notary service. *arXiv preprint arXiv:1804.00875*, 2018.

[352] P. Szalachowski. (Short Paper) Towards More Reliable Bitcoin Timestamps. In *IEEE CVCBT*. IEEE, 2018.

[353] P. Szalachowski, D. Reijsbergen, I. Homoliak, and S. Sun. Strongchain: Transparent and collaborative proof-of-work consensus. In *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019.*, pages 819–836, 2019.

[354] R. Tahir, M. Huzaifa, A. Das, M. Ahmad, C. Gunter, F. Zaffar, M. Caesar, and N. Borisov. Mining on someone else's dime: Mitigating covert mining operations in clouds and enterprises. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 287–310. Springer, 2017.

[355] W. Tang. Ecip-1029: Include uncles in total difficulty calculation, 2017.

[356] Team Rocket. Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies, 2018.

[357] P. Technologies. The multi-sig hack: A postmortem, 2017.

[358] S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov. SmartCheck: Static analysis of Ethereum smart contracts. In *WETSEB*. IEEE, 2018.

[359] A. Tobin and D. Reed. The inevitable rise of self-sovereign identity, 2016.

[360] A. Tomescu and S. Devadas. Catena: Efficient non-equivocation via Bitcoin. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 393–409, May 2017.

[361] C. F. Torres, J. Schütte, and R. State. Osiris: Hunting for integer bugs in Ethereum smart contracts. In *Proceedings of the 34th Annual Computer Security Applications Conference, ACSAC 2018, San Juan, PR, USA, December 03-07, 2018*, pages 664–676. ACM, 2018.

[362] trailofbits. Awesome Ethereum security, 11 Aug 2018.

[363] M. Tran et al. A stealthier partitioning attack against Bitcoin peer-to-peer network, 2019.

[364] Trezor. Trezor, 2018.

[365] True Names LTD. Ethereum name service. https://ens.domains/, 2019.

[366] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli, and M. Vechev. Securify: Practical security analysis of smart contracts. In *ACM CCS*. ACM, 2018.

[367] F. Tschorsch and B. Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):2084–2123, 2016.

[368] Unchained Capital. TrezorMultisig2of3: Ethereum Multisignature smart contract, 2018.

[369] Uniswap team. Uniswap: a protocol for automated token exchange on Ethereum, 2019.

[370] L. Valenta and B. Rowan. Blindcoin: Blinded, accountable mixes for bitcoin. In M. Brenner, N. Christin, B. Johnson, and K. Rohloff, editors, *Financial Crypto*, Berlin, Heidelberg, 2015.

[371] N. van Saberhagen. Cryptonote v 2.0, 2013.

[372] W3C community. Decentralized Identifiers (DIDs), 2019.

[373] W. Wang, D. T. Hoang, Z. Xiong, D. Niyato, P. Wang, P. Hu, and Y. Wen. A survey on consensus mechanisms and mining management in blockchain networks. *arXiv preprint arXiv:1805.02707*, pages 1–33, 2018.

[374] E. Ward. Breaking randomness in the Ethereum universe, 2018.

[375] W. Warren and A. Bandeali. 0x: An open protocol for decentralized exchange on the ethereum blockchain. *URl: https://github.com/0xProject/whitepaper*, 2017.

[376] J. Wilcke. The Ethereum network is currently undergoing a DoS attack, 2016.

[377] Z. Wilcox-O'Hearn. Names: Decentralized, secure, human-meaningful: Choose two. https://web.archive.org/web/20011020191610/http://zooko.com/distnames.html, 2003.

[378] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin. Storj a peer-to-peer cloud storage network, 2014.

[379] Wolfie Zhao. Bithumb $31 Million Crypto Exchange Hack: What We Know (And Don't), 2018.

[380] J. F. Wolfswinkel, E. Furtmueller, and C. P. Wilderom. Using grounded theory as a method for rigorously reviewing literature. *European journal of information systems*, 22(1):45–55, 2013.

[381] K. Wüst and A. Gervais. Ethereum eclipse attacks. Technical report, ETH Zurich, 2016.

[382] V. Wüstholz and M. Christakis. Harvey: A greybox fuzzer for smart contracts. *CoRR*, abs/1905.06944, 2019.

[383] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou. A survey of distributed consensus protocols for blockchain networks. *arXiv preprint arXiv:1904.04098*, 2019.

[384] Z. Xiao and Y. Xiao. Security and privacy in cloud computing. *IEEE Communications Surveys & Tutorials*, 15(2), 2013.

[385] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander. Where is current research on blockchain technology? – a systematic review. *PloS one*, 11(10):e0163477, 2016.

[386] J. Young. Analyst: Suspicious Bitcoin mempool activity, transaction fees spike to $16, 2017.

[387] J. Young. Analyst: Suspicious Bitcoin mempool activity, transaction fees spike to $16, 2017.

[388] M. Zamani, M. Movahedi, and M. Raykova. Rapidchain: Scaling blockchain via full sharding. In *ACM CCS*, 2018.

[389] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt. Sok: Communication across distributed ledgers. *IACR Cryptology ePrint Archive*, 2019:1128, 2019.

[390] A. Zamyatin, N. Stifter, P. Schindler, E. Weippl, and W. J. Knottenbelt. Flux: Revisiting Near Blocks for Proof-of-Work Blockchains, 2018. https://eprint.iacr.org/2018/415/20180529:172206.

[391] ZenCash. Zencash statement on double spend attack, 2018.

[392] Zeppelin Solutions. Serpent compiler audit, 2017.

[393] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi. Town Crier: An authenticated data feed for smart contracts. In *ACM CCS*. ACM, 2016.

[394] R. Zhang and B. Preneel. Publish or perish: A backward-compatible defense against selfish mining in Bitcoin. In *CT-RSA*. Springer, 2017.

[395] R. Zhang and B. Preneel. Lay down the common metrics: Evaluating proof-of-work consensus protocols' security. In *IEEE S&P*. IEEE, 2019.

[396] R. Zhang, R. Xue, and L. Liu. Security and privacy on blockchain. *arXiv preprint arXiv:1903.07602*, 2019.

[397] W. Zhang, Y. Yuan, Y. Hu, S. Huang, S. Cao, A. Chopra, and S. Huang. A privacy-preserving voting protocol on blockchain. In *11th IEEE International Conference on Cloud Computing, CLOUD 2018, San Francisco, CA, USA, July 2-7, 2018*, pages 401–408, 2018.

[398] K. Zhao, S. Tang, B. Zhao, and Y. Wu. Dynamic and privacy-preserving reputation management for blockchain-based mobile crowdsensing. *IEEE Access*, 7:74694–74710, 2019.

[399] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018.

[400] Y. Zhou, D. Kumar, S. Bakshi, J. Mason, A. Miller, and M. Bailey. Erays: reverse engineering Ethereum's opaque smart contracts. In *USENIX Security*, 2018.

[401] L. Zhu, B. Zheng, M. Shen, S. Yu, F. Gao, H. Li, K. Shi, and K. Gai. Research on the security of blockchain data: A survey. *arXiv preprint arXiv:1812.02009*, 2018.

[402] J. H. Ziegeldorf, R. Matzutt, M. Henze, F. Grossmann, and K. Wehrle. Secure and anonymous decentralized Bitcoin mixing. *Future Generation Computer Systems*, 80, 2018.

[403] ZILLIQA Team. The ZILLIQA Technical Whitepaper, 2017.

[404] H. Zimmermann. OSI reference model-the ISO model of architecture for open systems interconnection. *IEEE Transactions on communications*, 28(4), 1980.

[405] P. R. Zimmermann. *The official PGP user's guide*. MIT press, 1995.

[406] G. Zyskind, O. Nathan, and A. Pentland. Enigma: Decentralized computation platform with guaranteed privacy. *arXiv preprint arXiv:1506.03471*, 2015.