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

Ivan Homoliak, *Member, IEEE*, Sarad Venugopalan, Daniël Reijsbergen, Qingze Hum, Richard Schumi, and Pawel Szalachowski

***Abstract*—Blockchains are distributed systems, in which security is a critical factor for their success. However, despite their increasing popularity and adoption, there is a lack of standardized models that study blockchain-related security threats. To fill this gap, the main focus of our work is to systematize and extend the knowledge about the security and privacy aspects of blockchains and contribute to the standardization of this domain.We propose the security reference architecture (SRA) for blockchains, which adopts a stacked model (similar to the ISO/OSI) describing the nature and hierarchy of various security and privacy aspects. The SRA 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, and countermeasures, while we also analyze several cross-layer dependencies. Next, to enable better reasoning about security aspects of blockchains by the practitioners, we propose a blockchain-specific version of the threat-risk assessment standard ISO/IEC 15408 by embedding the stacked model into this standard. Finally, we provide designers of blockchain platforms and applications with a design methodology following the model of SRA and its hierarchy.**

***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 [1] or still under development [2], [3]. 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 [4], [5]. All of these areas are challenging, and it might take years until they are standardized and agreed upon 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 (SRA) for blockchains, which is based on models that demonstrate the stacked hierarchy of different threat categories (similar to the ISO/OSI hierarchy [6]) and is inspired by security modeling performed in the cloud computing [7], [8]. As our next contribution, we enrich the threat-risk assessment standard ISO/IEC 15408 [9] to fit the blockchain infrastructure. We achieve this by embedding the stacked model into this standard.

This article is based on our previous work outlining the security reference architecture [10]. 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.

Manuscript received October 22, 2019; revised August 4, 2020; accepted September 20, 2020. Date of publication October 26, 2020; date of current version February 24, 2021. This work was supported in part by the NRF, Prime Minister's Office, Singapore, through its National Cybersecurity Research and Development Programme under Award NRF2016NCR-NCR002-028, and administered by the National Cybersecurity Research and Development Directorate, in part by the Czech Science Foundation under Project 20-07487S, in part by the H2020 ECSEL Project VALU3S under Grant 876852, and in part by the Internal Project of Brno University of Technology under Grant FIT-S-20-6427. *(Corresponding author: Ivan Homoliak.)*

Ivan Homoliak is with the Faculty of Information Technology, Brno University of Technology, 601 90 Brno, Czech Republic (e-mail: ihomoliak@fit.vutbr.cz).

Sarad Venugopalan, Daniël Reijsbergen, Qingze Hum, Richard Schumi, and Pawel Szalachowski are with the iTrust, Center for Research in Cyber Security, Singapore University of Technology and Design, Singapore.

Digital Object Identifier 10.1109/COMST.2020.3033665

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

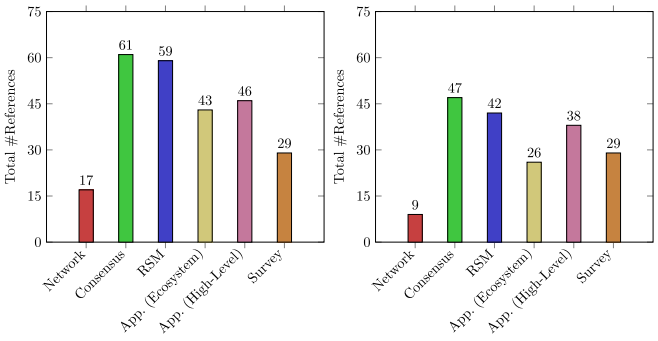
- For each layer, we model particular attacks or their categories through vulnerability/threat/defense graphs, while we provide reasoning about several important relations and causalities in these graphs.
- We modify and significantly extend the application layer, where we propose a novel functionality-oriented categorization, as opposed to the application-domain-oriented categorizations presented in related work [11], [12].
- 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.
- In the lessons learned, we describe the hierarchy of security dependencies among particular categories, based on which, we propose a methodology for designers of blockchain platforms and applications.

The rest of the paper is organized as follows: We describe the scope and methodology of our research as well as quantitative analysis of the included literature 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, 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. In Section X, we elaborate on lessons learned and propose a methodology for designers of blockchain-based solutions. We discuss the limitations of our work in Section XI and we compare our research to related work in Section XII. Finally, we conclude the paper in Section XIII.

II. Methodology & Scope

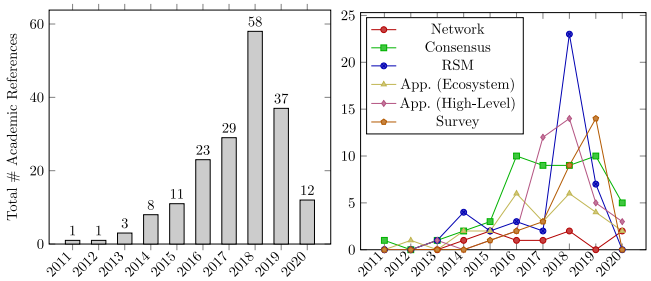
In contrast to conventional survey approaches, such as grounded theory for rigorous literature review [13], we do not sample included research from existing databases queried with specific terms. Instead, we study and analyze security-oriented literature for vulnerabilities and threats related to the blockchain infrastructure. The literature that we select as the input mainly covers existing blockchain-oriented surveys as well as various conferences and journals in security and distributed computing. Moreover, we include other materials published in preprints, whitepapers, products' documentation, and forums, which are also related.

We aim to consolidate the literature, categorize found vulnerabilities and threats according to their origin, and as a 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. See the road-map of all the categories in Figure 4. Our next aim is to indicate and explain the



(a) All references (b) Academic references

Fig. 1. Blockchain-specific references per category.



(a) #References per year (b) #References per year/category

Fig. 2. Blockchain-specific academic references over time.

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/crypto-jacking [14]) 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.

A. Quantitative Analysis

For a quantitative summary of the literature, we considered only the references from the main sections that correspond to particular layers of the proposed stacked model (i.e., Section V to Section IX) and related work (Section XII). We excluded references from other sections and the references on the examples of incidents (Appendix C). Each assumed reference was labeled with three flags to indicate:

- The category to which it belongs. Note that some papers belong to multiple categories.
- Whether it is specifically related to blockchains.
- Whether it was written by academics (i.e., such that their affiliation was listed in the document). Note that the

positive value of this flag covers also the papers that have not been peer-reviewed but which appeared on public repositories (e.g., arXiv and Cryptology ePrint Archive).

In sum, we collected 276 references, of which 247 were blockchain-specific, 211 were written by academics, and 180 met both criteria. In Figure 1, we show the number of references in each of the main categories for (a) all papers and (b) the academic papers. A total of 11 papers were found to belong to two categories. We observe that most of the references are presented at the application layer, which we conjecture that occurred because each application running on the blockchain might introduce new attack vectors.

The evolution of the number of references over time is depicted in Figure 2, where we observe that the number of academic papers about blockchains and security has increased steadily, although 2018 was the year with the highest number of references selected for the current work.[1] Of the four layers discussed in this article, the network layer has the lowest number of blockchain-focused papers. A possible reason for it could be that among the remaining layers, the consensus and replicated state machine (RSM) layers are more specific to blockchains, while the application layer is popular among practitioners building on top of other layers. Finally, we observe the number of surveys is steadily growing (culminating in 2019), which indicates increasing interest in this domain.

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.

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 a native crypto-token. Nevertheless, other blockchain applications have emerged, benefiting from features other than decentralization, e.g., privacy, energy efficiency, throughput, etc. For the review of the inherent and non-inherent features of the blockchains, we refer the reader to Appendix A.

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 3):

[1]Note that we started to work on this survey in early 2019.

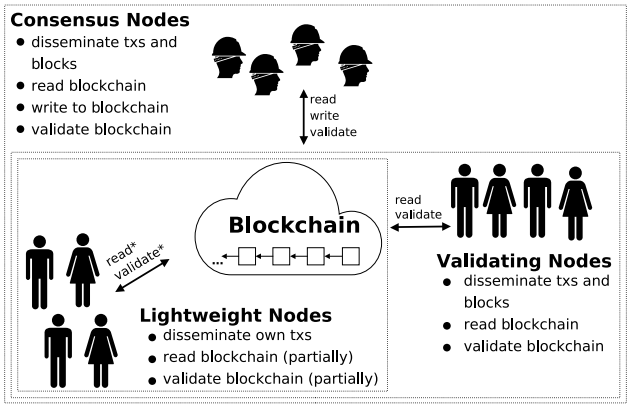


Fig. 3. Involved parties with their interactions and hierarchy.

(1) Consensus nodes (a.k.a., *miners* in Proof-of-Resource protocols) 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. Consensus nodes can prevent malicious behaviors (e.g., by not appending invalid transactions, or ignoring an incorrect chain).

(2) Validating nodes read the entire blockchain, validate it, and disseminate transactions. Unlike consensus nodes, validating nodes cannot write to the blockchain, and thus they cannot prevent malicious behaviors. On the other hand, they can detect malicious behavior since they possess copies of the entire blockchain.

(3) Lightweight nodes (a.k.a., clients or Simplified Payment Verification (SPV) clients) benefit from most of the blockchain functionalities, but they are equipped only with limited information about the blockchain. These nodes can read only fragments 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. Therefore, they can detect only a limited set of attacks, pertaining to their own transactions.

Additional Involved Parties: Note that besides native types of involved parties, many applications using or running on the blockchain introduce their own (centralized) components.

B. 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. To prevent Sybil attacks, this type of blockchains usually requires consensus nodes to establish their identities by running a Proof-of-Resource protocol, where the consensus power of a node is proportional to its resources allocated.

Permissioned blockchains require a consensus node to obtain permission to join the consensus protocol from a centralized or federated authority(ies), while nodes usually have equal consensus power (i.e., one vote per node).

Semi-Permissionless blockchains require a consensus node to obtain some form of permission (i.e., stake) before joining the protocol; however, such permission can be given by

any consensus node. The consensus power of a node is proportional to the stake that it has.

C. Design Goals of Consensus Protocols

1) Standard Design Goals–Liveness and Safety: **Liveness** ensures that all valid transactions 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. Usually, consensus protocols satisfy safety and liveness only under certain assumptions: the minimal fraction of honest consensus power or the maximal fraction of adversarial consensus power. With regard to safety, literature often uses the term *finality* and *time to finality*. **Finality** represents the sequence of the blocks from the 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 need to be appended after B – the number of such blocks is referred to as *the number of confirmations*.

2) Specific Design Goals: As a result of this study, we learned that standard design goals of the consensus protocol should be amended by specific design goals related to the type of the blockchain. In permissionless type, *elimination of Sybil entities*, *a fresh and fair leader/committee election*, and *non-interactive verification of the consensus result* is required to meet. In contrast, the (semi)-permissionless types do not require the elimination of Sybil entities (see details in Section X-C).

3) Means to Achieve Design Goals:

a) Simulation of the verifiable random function (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 can be captured by the concept of the VRF [15], which ensures the unpredictability and fairness of the election process. Therefore, the leader/committee election process can be viewed as a simulation of VRF [16]. Due to the properties of VRF, the correctness of the election result can be verified non-interactively after the election took place.

b) 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) blockchains that introduce their native crypto-tokens, this is achieved by 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., [17]), although alternative incentive schemes rewarding more consensus nodes at the same time are also possible (e.g., [18]). While transaction fees are included in a particular transaction, the block reward is usually part of the first transaction in the block (a.k.a., *coinbase* transaction).

D. Basis of Consensus Protocols

Lottery and *voting* are two marginal techniques that deal with the establishment of a consensus [19]. However, in addition to them, their combinations have become popular.

Lottery-Based Protocols: These protocols provide consensus by running a lottery that elects a leader/committee, who produces the block. The advantages of lottery-based approaches are a small network traffic overheads and high scalability since the process is usually non-interactive (e.g., [17], [20], [21]). However, a disadvantage of this approach is the possibility of multiple "winners" being elected, who propose conflicting blocks, which naturally leads to inconsistencies called *forks*. Forks are resolved by fork-choice rules, which compute the difficulty of each branch and select the one. For the *longest chain rule*, the chain with the largest number of blocks is selected in the case of a conflict, while for the *strongest chain rule*, the selection criteria involve the quality of each block in the chain (e.g., [18], [22], [23], [24], [25]). Note that the possibility of forks in this category of protocols causes an increase of the time to finality, which in turn might enable some attacks such as double-spending.

Voting-Based Protocols: In this group of protocols, the agreement on transactions is reached through the votes of all participants. Examples include Byzantine Fault Tolerant (BFT) protocols – which require the consensus of a majority quorum (usually (2/3) of all consensus nodes (e.g., [26], [27], [28], [29], [30]). The advantage of this category is a low-latency finality due to a negligible likelihood of forks. The protocols from this group suffer from low scalability, and thus their throughput forms a trade-off with scalability (i.e., the higher the number of nodes, the lower the throughput).

Combinations: To improve the scalability of voting-based protocols, it is desirable to shrink the number of consensus nodes participating in the voting by a lottery, so that only nodes of such a committee vote for a block (e.g., [29], [31], [32], [33], [34]). Another option to reduce active voting nodes is to split them into several groups (a.k.a., *shards*) that run a consensus protocol in parallel (e.g., [35], [36]). Such a setting further increases the throughput in contrast to the single-group option, but on the other hand, it requires a mechanism that accomplishes inter-shard transactions.

E. Failure Models in Distributed Consensus Protocols

The relevant literature mentions two main failure models for consensus protocols [37]:

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 [38], Raft [39], Viewstamped Replication [40]).

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 [17], pure BFT protocols [26], [27], [28], [41], and hybrid protocols [31], [35], [36]).

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 applications 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 the security aspects of blockchains, we utilize a stacked model consisting of four layers (see Figure 4). A similar stacked model was already proposed in the literature [16], but in contrast to it, we preserve only such a granularity level that enables us to isolate security threats and their nature, which is the key focus of our work. In the following, we briefly describe each layer.

(1) The network layer 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 deals with the ordering of transactions, and we divide it into three main categories according to the protocol type: Byzantine Fault Tolerant, Proof-of-Resource, and Proof-of-Stake protocols.

(3) The replicated state machine (RSM) layer deals with the interpretation of transactions, according to which the state of the blockchain is updated. In this layer, transactions are categorized into two 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 the 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 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 of application types resides at a 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 15).

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.

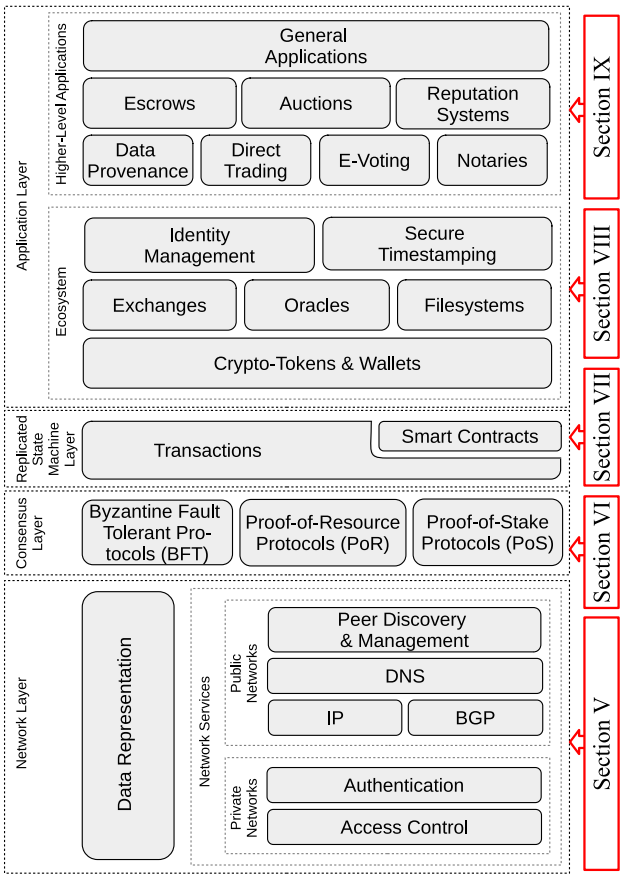


Fig. 4. Stacked model of the security reference architecture.

B. Threat-Risk Assessment Model

To better capture the security-related aspects of blockchain systems, we introduce a threat-risk model (see Figure 5) that is based on the template of ISO/IEC 15408 [9] and projection of our stacked model (see Figure 4). This model includes the following components and actors:

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

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

Threat agents are spread across all the layers of the stacked model, and they mostly involve 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, and they exist at all layers of the stacked model. Threats arise from vulnerabilities in the network, smart contracts, applications, from consensus protocol deviations, violations of consensus protocol assumptions.

Countermeasures protect owners from threats by minimizing the risk of compromising/losing the assets. Alike the threats and threat agents, countermeasures can be applied at each of the layers of our stacked model, and they involve various security/privacy/safety solutions, incentive schemes, reputation techniques, best practices, etc. Nevertheless, we emphasize that their utilization usually imposes some limitations such as higher complexity and additional performance overheads (e.g., resulting in decreased throughput).

Risks are related to the application layer, and they are caused by threats and their agents. Risks 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 the 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.

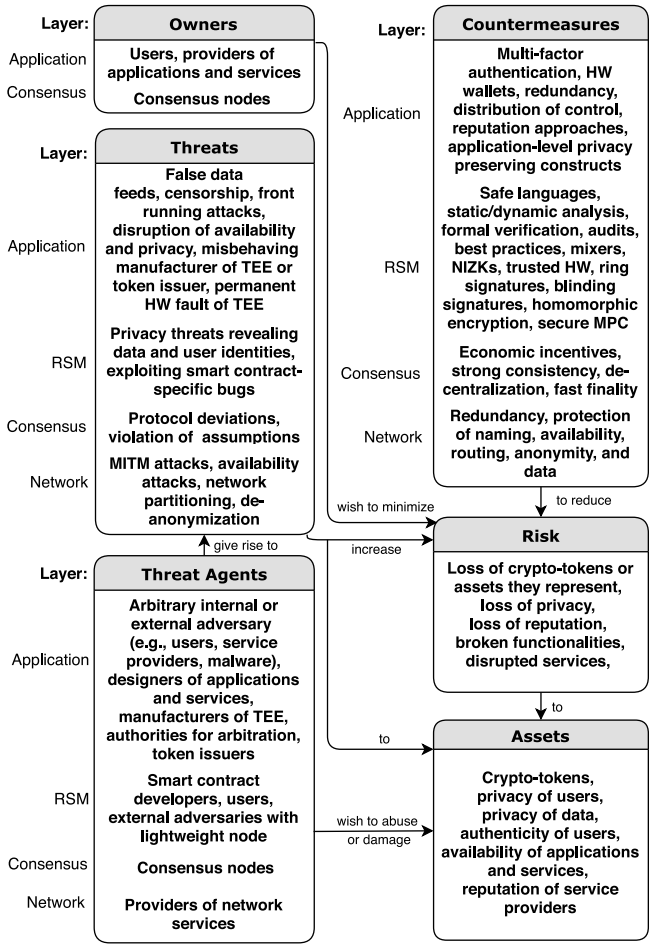


Fig. 5. Threat-risk assessment model of the security reference architecture.

Involved Parties & Blockchain's Life-Cycle: In Section III, we presented several types of involved parties in the blockchain infrastructure (see Figure 3). 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

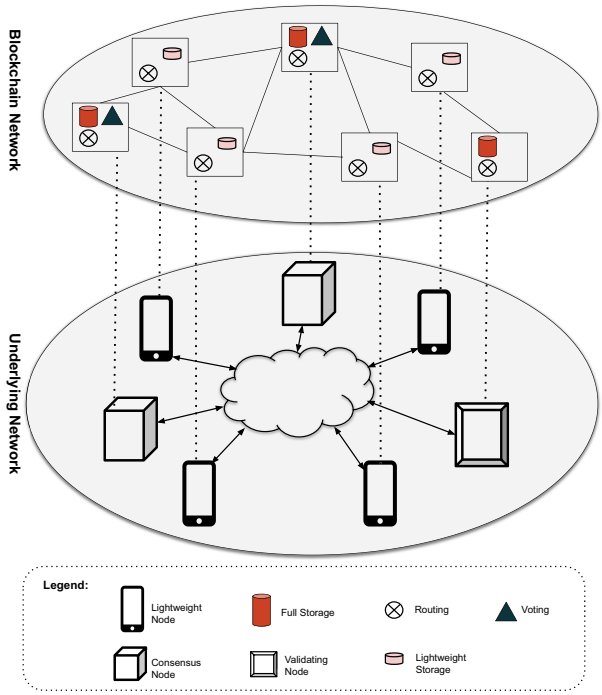


Fig. 6. Overlaying of blockchains over a private/public network.

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 thoroughly reviewed by the community or public, as it is typical for other (lower) layers.

V. Network Layer

Blockchains usually introduce peer-to-peer overlay networks built on top of other networks (see Figure 6). Hence, blockchains inherit security and privacy issues from their underlying networks. In our model (see Figure 4), we divide the network layer into *data representation* and *network services* sub-planes. The data representation plane is protected by cryptographic primitives that ensure data integrity, user authentication, and optionally confidentiality, privacy, anonymity, non-repudiation, and accountability. The main 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. Based on permission to join the blockchain system, the networks are either private or public. In the following, we discuss the pros and cons of private and public networks as well as their security threats and mitigation techniques.

A. Private Networks

A private network ensures low latency, a centralized administration, privacy, and meeting regulatory obligations (e.g., HIPAA[2] 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. Private networks inherently provide authentication and access control.

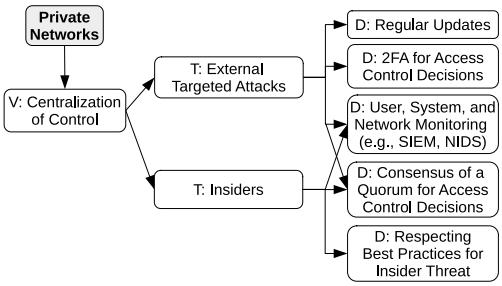


Fig. 7. Vulnerabilities, threats, and defenses in private networks (network layer).

1) Pros: *Access control* is achieved by centralized authentication of users. A private network has full control over routing paths and physical resources used, which enables suitable regulation of the network topology with regard to the given requirements. *Data privacy* is ensured by permissioned settings. *User identities* might only be revealed within a private group of nodes. *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.

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. Private networks are suitable only for permissioned blockchains.

3) Security Threats and Countermeasures: We present a taxonomy of vulnerabilities, threats, and defenses against them in Figure 7. We identified *insiders* and *external targeted attacks* as the specific security threats for the (permissioned) blockchains running over the private networks. These threats are possible due to a centralization of access control that might occur in private networks, and thus permissioned blockchains. An external attacker might exploit a network or system vulnerability and obtain access to an element responsible for access control to the blockchain. In the case of the insider threat, she may already have the necessary privileges or obtain them by exploiting the system, network, or organization vulnerabilities. As a result, the insider might add malicious consensus nodes into the network or remove legitimate ones, and thereby increase the adversarial consensus power that is manifested at the consensus layer. In turn, this may lead to plenty of attacks occurring at the consensus layer

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

(see Section VI) such as attacks on violation of protocol assumptions.

Countermeasures include regular software updates, monitoring of users, network, and systems (e.g., by SIEM, anti-virus software, intrusion detection systems), prevention techniques that minimize trust and maximize trustworthiness such as two-factor authentication for access control decisions (effective for the external attacker), as well as respecting best practices for mitigation of insider threat [42]. Another option of coping with the centralization of access control is to embed decentralized access control into the consensus layer and thus mandating a requirement on reaching a consensus of a quorum of nodes for each access control decision. In contrast to the other mentioned countermeasures, the embedding of access control into the consensus layer is more effective since it eliminates a single-point-of-failure of centralized access control service, and thus it makes an increase of adversarial consensus power more difficult for the attacker – the attacker has to compromise a high number of independent consensus nodes.

4) Side Effects of Countermeasures: Most of the mentioned countermeasures do not cause negative effects on the features of the blockchains under normal circumstances. The exception might be embedding of access control into the consensus layer under the assumption that the set of consensus nodes in the network is extremely dynamic, and thus nodes are entering and leaving the blockchain very often. In such a case, the throughput of the blockchain might be decreased.

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. *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.

2) Cons: Single-point-of-failure – DNS with its hierarchy, IP addresses, and autonomous systems (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 to [43]: (1) resources under attacker control (e.g., botnets, DNS and BGP servers), (2) stolen or masqueraded identities (e.g., IP addresses participating in an eclipse attack or route manipulation), (3) MITM attacker (i.e., eavesdropping and spoofing), (4) the exploitation of common network vulnerabilities, (5) revealing secrets (e.g., de-anonymizing peers). *Efficiency* – although an average Internet bandwidth has been improved in recent years, distribution of powerful infrastructure is not uniform, which results in a different latency among peers, and thus the overall latency of the network is increased; this might result in the loss of created blocks and thus wasting consensus power.

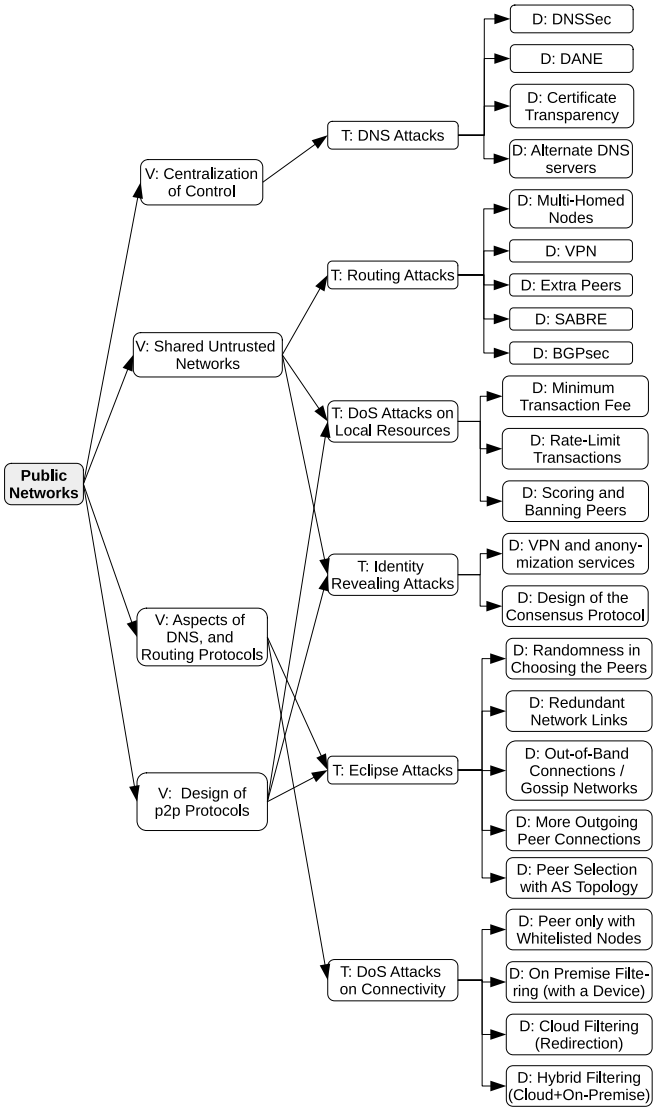


Fig. 8. Vulnerabilities, threats, and defenses in public networks (network layer).

3) Security Threats and Countermeasures: We present a taxonomy of vulnerabilities, threats, and defenses related to public networks in Figure 8, while in Table VI of Appendix, we list several incidents that occurred in practice. In the following, we describe these threats as well as possible defense techniques.

DNS attacks arise from cache poisoning that mainly affects blockchains employing centralized DNS bootstrapping to retrieve online peers from a hard-coded list of DNS seeders. 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 [44].

Routing attacks are traffic route diversions, hijacking, or DoS attacks. Besides 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 presented at the consensus layer (see Section VI). Apostolaki *et al.* [45] demonstrated that the Bitcoin protocol is vulnerable to BGP routing attacks where the attacker controlling a transit autonomous system (AS) can modify inter-domain routes for a few Bitcoin nodes and cause network partitioning. However, perpetration of this attack reveals the identity of the malicious AS, which might have immediate reputation consequences.

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 [45]. Another mitigation is SABRE [46], a secure relay network that runs alongside with the Bitcoin network. Further, BGPsec is a security extension for BGP used between neighboring ASes, and it assures route origin and propagation by cryptographic verification.

Eclipse attacks aim to hijack all connections of a node to its peers in a 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 [47], [48], [49]. Eclipse attacks increase chances of selfish mining and double-spending attacks (see Section VI) – the eclipsed victims may unknowingly vote for an attacker's chain, and thus cause a network partitioning. Erebus [50] is a stealthier attack causing network partitioning as compared to Apostolaki *et al.* [45]. However, Erebus is not a routing attack since it does not involve BGP prefix hijacking (which is easy to detect) and has a very small network traffic footprint. In Erebus, the attacker controls a large number of shadow IP addresses and influences the victim's peer selection mechanism to pick all outgoing connections with shadow IP addresses. This is achieved through slowly flooding the victim's peer tables by incoming connections from the attacker-controlled shadow IP addresses.

Countermeasures: Improving randomness in choosing peers was proposed in the work of Heilman *et al.* [47] 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). Eclipse attacks can also be detected by employing out-of-band gossip networks (e.g., Web-servers) [51] that communicate with lightweight clients to exchange their views on the blockchain (i.e., block headers) along with native Web traffic. Erebus attacks can be made much harder by decreasing the size of the peer tables, increasing the number of peers, preferring the peers that provide fresh data, and incorporation the topology of ASes into the peer selection process. 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 [52]. For validating nodes, this attack leads to a disruption of some blockchain-dependent services [53]. *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 [54].

DoS attacks on resources such as memory and storage, may reduce the peering and consensus capabilities [55] 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 were applied to Bitcoin [56] nodes including scoring DoS attacks and banning misbehaving peers. DoS attacks on connectivity 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.

Identity revealing attacks are conducted by linking the IP address of a node with an identity propagated in transactions [57], [58]. Traffic analysis using Sybil listeners can reveal the linkage of node IP addresses and their transactions [59]. *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.

4) Side Effects of Countermeasures: The anonymization services cause deterioration of connectivity for consensus nodes, and these nodes might not distribute the created block on time and thus lose their reward. On the other hand, the slow connectivity of anonymization services can be acceptable for validating nodes and clients transmitting messages related to the creation and validation of their transactions. The trade-off for connectivity and anonymity can be provided by VPN services, which are fast but they are usually operated by a centralized party, and thus the risk of de-anonymization is higher than in the case of anonymization services. Increasing the number of outgoing peer connections as protection against eclipse attacks [50] can increase the volume of data that needs to be transferred, which might negatively impact the throughput of blockchains. Furthermore, fetching the same blocks

from multiple peers [45] may contribute to the network congestion under certain configurations of blockchains focusing on a high throughput – this might lead to a slow down in keeping touch with the tip of the blockchain and thereby impact a response time of an application running on top of it. The response time of the application might be also impacted by cloud and hybrid filtering that relay some incoming traffic through 3rd party services.

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 concerning different principles of operation and thus their security aspects. First, we focus on the security aspects that are generic to all categories, and then we detail each category.

A. Generic Attacks

We present a taxonomy of the generic threats to all types of consensus protocols, their origins, and defenses against them in Figure 9. These threats originate mainly from *violation of protocol assumptions* but also due to *a long time to the finality* of some consensus protocols. In the following, we describe these threats as well as possible defense techniques.

1) Security Threats and Mitigations:

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 her control. In *Byzantine attacks*, a quorum of $\frac{1}{3}$ adversarial consensus nodes might cause the protocol to be disrupted or even halted. As a design-oriented countermeasure, it is important to promote decentralization by incentive schemes that reward honest participation and discourage [60] or punish [32], [61] protocol violations. Another mitigation that makes these attacks more expensive is a statistical analysis of sudden anomalies in the history of the consensus power distribution among nodes, which can be embedded in the fork-choice rule of the consensus protocol [62].

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 [63]. This fact motivates asynchronous BFT protocols that can be based on threshold-based cryptography, which enables reliable and consistent broadcast [41], [63].

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 [64]. 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 [65]. Another option to improve the accuracy of block timestamps is to compute them collaboratively by consensus nodes [18].

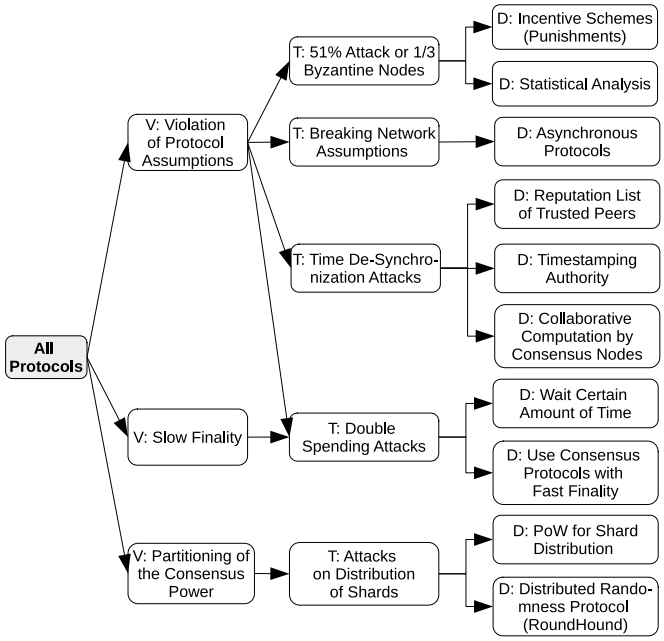


Fig. 9. Generic threats and defenses of the consensus layer.

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% attacks.[3] To prevent this attack, it is recommended to wait a certain amount of time (i.e., time to the finality) until a block "is settled" or utilize consensus protocols with fast time to the finality, such as BFT protocols and their combinations.

Attacks on Shards: Sharding means that consensus nodes are distributed among subgroups (i.e., shards) such that each node only validates the transactions in its group. Shards operate in parallel and can achieve higher scalability and throughput since each shard has a throughput similar to an entire non-sharded blockchain. On the other hand,

[3]Note that in the case of PoR protocols, this attack may also occur as a consequence of the selfish mining attack.

sharding has the potential to harm security because each shard has a lower number of participating nodes than the entire blockchain, which means that it may be easier for the attacker to compromise a single shard than the entire blockchain [66], [67]. The main mitigation technique is to achieve a truly random distribution of nodes among the shards, and thus minimize the potential for adversaries to bias the randomness used for shard distribution. For example, Elastico [68] uses PoW to distribute nodes among shards, whereas Omniledger [35] uses a bias-resistant distributed randomness protocol (e.g., RandHound [69]). Poor design of sharding protocols may also lead to vulnerabilities such as replay attacks [70].

2) Side Effects of Countermeasures: Some generic countermeasures presented in the above text might impact the features of the blockchains. For example, the application layer countermeasure that uses the timestamping authority brings centralization issues, which might impact the availability of the service and enable misbehaving of the authority that might provide imprecise time. To cope with this issue, instead of using the application layer countermeasure, the computation of block timestamps can be embedded into the consensus protocol, e.g., by ensuring that multiple consensus nodes simultaneously contribute to global time using partial solutions [18].

Enriching a fork-choice rule by statistical analysis of the history [62] might protect against sudden changes in the distribution of the overall consensus power (e.g., rent-and-attack in PoW protocols), but not against a slow gradual increase of the consensus power by a coalition. Furthermore, this mitigation technique incentivizes consensus nodes to use stable identifications (i.e., the same key pairs) to increase the strength of the honest chain and thus the chances that it becomes the main chain after a temporary fork. This might impact the privacy and security of consensus nodes, who are disincentivized to rotate keys. Note that some privacy issues can be resolved at the RSM layer (see Section VII).

Using a BFT consensus protocol helps to significantly decrease the likelihood of double-spending attacks, but on the other hand, it worsens the scalability of the blockchain and thus throughput, especially in the case of a high number of consensus nodes. These issues can be resolved by combining a BFT voting protocol with lottery-based protocols that reduce the size of the actively communicating nodes to a small committee (e.g., [31], [32], [33], [34], [29]). The scalability of BFT protocols can be also improved by using threshold-based signature aggregation with a gossip-based communication pattern (e.g., [71], [72]).

Distributed randomness protocols and PoW for distribution of shards bring additional overheads, which may reduce the throughput of the blockchain; however, this reduction is negligible in contrast to the throughput improvement due to sharding.

B. Proof-of-Resource Protocols (PoR)

Protocols from this category require nodes to prove the spending of a scarce resource in a lottery-based fashion [19]. 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 [73] (e.g., Spacecoin [74], SpaceMint [75]). *(3) Crypto-tokens* spent for Proof-of-Burn protocols [76] (e.g., Slimcoin [77]). *(4) Combinations and modification* of the previous types, such as storage and computation, called Proof-of-Retrievability (e.g., Permacoin [78]) and storage over time, which is represented by Proof-of-Space protocols (e.g., Filecoin [79]). Another hybrid example of this category is a combination of PoR with elapsed time, such as in PeerCoin [80]. However, it is a philosophical question whether to 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 towards the second option.

PoR protocols belong to the first generation of consensus protocols, and they are mostly based on Nakamoto Consensus [17] that utilizes 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 [81].

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

2) Cons: PoR protocols imply high operational costs. 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 10, while we list several real-world incidents in Table V of Appendix. In the following, we describe these attacks as well as possible defense techniques.

Selfish Mining: In selfish mining [82],[4] 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 the consensus power of a selfish miner reaches some threshold (e.g., 30%). The selfish mining strategy was later generalized [84] and extended to other variants that increase the profit of the attacker [85]. *Countermeasures:* For the case of the longest-chain rule, the first introduced mitigation is uniform tie-breaking [82], 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 [84]. As the longest-chain rule enables this attack, it is recommended to use other fork-choice rules that also

[4]Note that selfish mining is theoretically possible even in PoS protocols [83] but requiring them to have predictable randomness for the leader election, which is usually a design-oriented vulnerability (see Section III-C). However, in PoR, selfish mining is possible even with unpredictable randomness for the leader election.

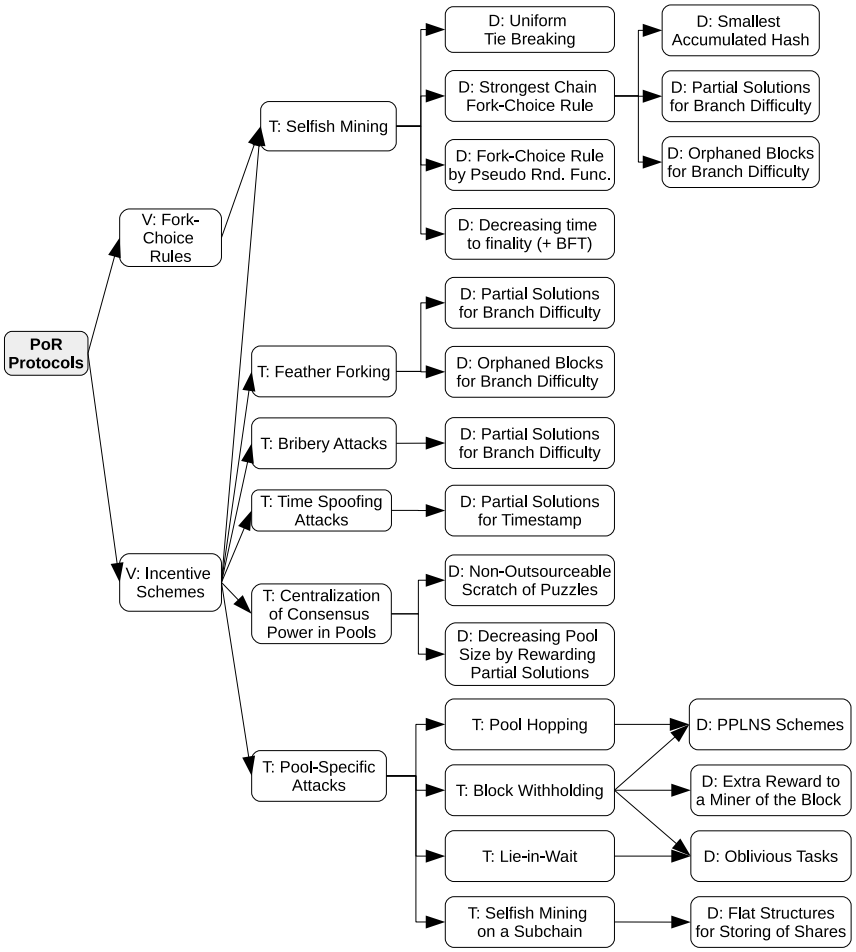


Fig. 10. Vulnerabilities, threats, and defenses of PoR protocols (consensus layer).

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 of the header. Another example is to include partial solutions [18], [25], [86] or solutions representing full (orphaned) blocks [23], [87] in the computation of a chain quality. These partial or orphaned solutions can be incentivized by rewards to further improve decentralization. Another option for a deterministic fork-choice rule is using a pseudo-random function [88], which moreover provides unpredictability,[5] hence an attacker cannot determine his chances to win a tie. Finally, PoW protocols can be combined with BFT protocols, where PoW is used only for joining the protocol and BFT for consensus itself (e.g., [34], [35], [36], [68], [88]).

Feather Forking: In this attack [89], 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 [18], [25], [86], [90] or full orphaned blocks [23], [87] 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 [91] or to reorder transactions within a block, and thus enable the transaction front-running by other means than natural priority gas auctions (PGAs) [92].[6]

Countermeasures: assuming that the miners who accept bribes constitute a minority, a possible mitigation technique is to utilize partial solutions [18], [25], [86], [90], which reduce the likelihood that the double-spending attacks succeed. Regarding "bribed" transaction front-running, the misbehavior happens entirely off-chain

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

[6]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)

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 crypto-tokens after these attacks are publicly disclosed).

Time-Spoofing Attacks: Time-spoofing attacks target a time-based difficulty computation algorithm in a PoR protocol with the intention to decrease the difficulty of the puzzle and thus minimize the 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 the accuracy of the timestamps may utilize partial solutions found by all nodes into an averaged timestamp computation [18]. Note that the impact of time spoofing attack might be significant also at the application layer, especially in use cases that rely on timestamp accuracy (see Section VIII-F).

Pool Specific Attacks: Since PoR protocols are usually based on a lottery having a single winner [17], rewards for participation impose 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 [60] avoid the 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 [18], [25], [86], [90] and thus minimize payout variance. For a detailed analysis of rewarding schemes in pools, we refer the reader to [93]. 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 [94] since 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 [95] can be used. PPLNS removes the concept of rounds and instead of immediate payments, it employs deterred payments after N shares are submitted 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 [93]. Such withholding does not mean a direct gain for the attacker, but she may do a secret agreement with concurrent pool(s) that may reward the attacker for showing a withheld block [96] (a.k.a., sponsored block withholding). Mitigation for this kind of attack is using the PPLNS scheme, giving an extra reward to the miner of the block [97], precluding miners from distinguishing between a share and a full solution (i.e., oblivious tasks).

c) Lie-in-wait: If the miner finds a block in a victim pool, she does not immediately submit it to the pool operator, but instead focuses all her available mining power on the victim pool to increase her relative shares within a pool; after some time attacker releases the formerly found block. A countermeasure for this attack is an oblivious task [96].

d) Selfish mining on a subchain: Decentralized mining pools, such as p2pool [98], achieve decentralization by updating an intermediary coinbase transaction with mined shares. To preserve consistency with the previous versions of the 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.[7] A possible countermeasure is to use flat data structures for aggregation of shares, such as the Merkle tree or hash of a set [18].

4) Side Effects of Countermeasures: Partial solutions for difficulty computation might cause additional network overheads and thus decrease the throughput of the protocol. However, this is not the case for sufficiently long rounds of PoW protocols, such as in StrongChain [18]. On the other hand, rewarding partial solutions, apart from mitigating some threats, helps to promote decentralization – payout variance is decreased and thus mining pools are not needed in some cases, and in other cases, they can be much smaller.

C. Byzantine Fault Tolerant (BFT) Protocols

BFT protocols represent voting-based [19] consensus protocols that utilize Byzantine agreement and a state machine replication [37]. These protocols assume a fully connected topology, broadcasting messages, and a master-replicas hierarchy. Synchronous examples of this category are PBFT [26], RBFT [27], eventually synchronous examples are BFT-SMaRt [99], Tendermint [28], Byzantine Paxos [100], BChain [30], and asynchronous examples are SINTRA [41] and HoneyBadgerBFT [63]. For more details, we refer the reader to a review of BFT protocols and their practical applications in both permissioned and permissionless blockchains [101].

1) Pros: BFT protocols provide high throughput and fast finality. Another advantage of BFT protocols is that they can be combined with PoS or PoR protocols to achieve reasonable scalability and retain their other properties. This is in line with a lottery approach [19] for selecting a portion of all nodes,

[7]Note that the same applies for Flux [25] and Subchains [90] that maintain a subchain but at the level of the whole network (as opposed to p2pool).

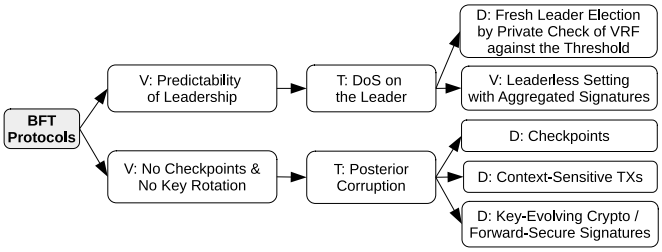


Fig. 11. Vulnerabilities, threats, and defenses of BFT protocols (consensus layer).

referred to as a committee, which further runs BFT consensus or its part (e.g., Algorand [31], Zilliqa [34], DFINITY [33]).

2) Cons: The main con of traditional BFT protocols [26], [100] is 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.

3) Improvements: The issues with scalability vs. throughput of BFT protocols can be partially addressed by applying threshold signatures (e.g., HotStuff [102], ByzCoin [88], Theta Blockchain [71]) as well as by optimizing communication patterns from original broadcast to gossiping [71] or communication trees [88]. On top of that, consensus nodes might be partitioned into shards that process transactions in parallel (e.g., Omniledger [35], RapidChain [36]).

4) Security Threats and Mitigations: We present a taxonomy of the attacks related to BFT protocols, their origins, and defenses against them in Figure 11. 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 the 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 might result in a restart of the round. *Countermeasures:* To prevent this attack, a node can privately determine whether it is a potential leader by using Verifiable Random Function (VRF) [31], and immediately release a block candidate; hence, after publishing this data, it is too late for a DoS attack on the node. Another option for coping with this attack can be implemented by aggregating threshold signatures in a leaderless setting [71].

Posterior Corruption: Posterior corruption is a specific instance of a violation of protocol assumptions (see Section VI-A) in which the adversarial consensus power reaches $\frac{2}{3}$. In posterior corruption, the adversary has to steal private keys of $\frac{2}{3}$ possibly "retired" consensus nodes and then rerun the consensus protocol, rewriting the history of the blockchain. Although this attack is mainly discussed in the context of PoS protocols (see Section VI-D), we note that in PoS the attacker's motivation is typically financial and exploits an incentive scheme of (semi-)permissionless blockchains. On the other hand, the attacker's motivation might be different in BFT protocols in permissioned settings (e.g., governance or sabotage) since many such BFT protocols do not contain an incentive scheme. To mitigate posterior corruption, key-evolving cryptography [103] and forward-secure digital signatures (e.g., d-ary certificate trees [104]) can be employed, requiring users to evolve their private keys and erase already used keys. Another option is to employ irreversible checkpoints after a fixed number of blocks or context-sensitive transactions, which put the hash of a recent valid block into a transaction itself [105].

5) Side Effects of Countermeasures and Improvements: A problem of some threshold signatures (e.g., BLS signatures) is a lack of forward secrecy, which might enable posterior corruption attacks. Forward secrecy might be provided by schemes such as d-ary certificate trees [104]. On the other hand, a disadvantage of some forward-secure digital signatures (e.g., [104]) is an extra overhead required for their verification and updating, which negatively influences the throughput of the blockchains. This problem was addressed in Pixel signatures [106], in which the authors proposed aggregated signatures supporting forward secrecy and demonstrated bandwidth and storage savings in contrast to d-ary certificate trees [104].

Pruning the number of nodes that run BFT into committees [31] reduces the security level of BFT and provides only probabilistic security guarantees depending on the committee size. This phenomenon might be also seen as a deterioration of decentralization.

D. Proof-of-Stake Protocols (PoS)

Similar to the PoR category, PoS protocols are based on the lottery approach [19]. However, in contrast to PoR, no scarce resource is spent; instead, the nodes are required "to prove investment" of crypto-tokens to participate in a protocol, and thus eventually earn interest from the invested amount. The concept of PoS was for the first time proposed in the Bitcointalk forum [107]. The first technical realization of PoS is Peercoin [80], 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 couple of pure PoS protocols (e.g., Chains of Activity [108], Ouroboros [21]), the trend is to combine them in a hybrid setting with PoR (e.g., Proof-of-Activity [109], Peercoin [80], Snow White [110]) or BFT protocols (e.g., Algorand [31], Theta Blockchain [71]). In particular, a combination of PoS with BFT represents a promising approach, which takes advantage of both lottery and voting (i.e., scalability and throughput), where 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., [80], [110]), the overall energy spent is much smaller than in the case of pure PoR protocols.

2) Cons: The introduction of PoS protocols has brought PoS specific issues and attacks, while these protocols are, at the time of writing, still not formally proven to be secure. Next, PoS protocols are semi-permissionless – a node needs

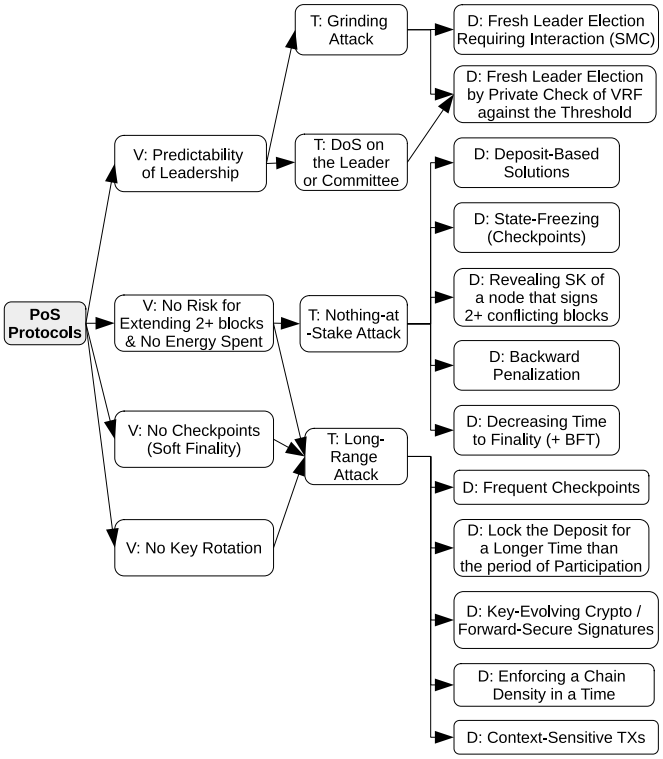


Fig. 12. Vulnerabilities, threats, and defenses of PoS protocols (consensus layer).

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 12. In the following, we describe these attacks as well as possible defense techniques.

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., [61]) require nodes to make a deposit during some fixed period/round and checkpoint-based solutions (e.g., [32], [61], [80], [111]) 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 [112] that enable anybody to reveal the 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 [32], [61]. Finally, PoS protocols can be combined with BFT approaches, and thus the probability of forks is negligible (e.g., [31]).

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 her 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 can be prevented by performing a fresh leader election by an interaction of consensus nodes within some committee (e.g., the secure multiparty coin-flipping protocol [21]) or by privately checking whether the VRF output is below a certain stake-specific threshold (e.g., [31]). The input of the VRF is the user's private key and the randomness unambiguously bound to the previous block; hence each consensus node might compute 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 [21], then the adversary may conduct a DoS attack against them and thus cause a restart of the round – this might be repeated until the adversary's desired nodes are elected. *Countermeasures:* A prevention technique was proposed in Algorand [31] – 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 the VRF was also utilized in other protocols (e.g., [33], [113]).

Long-Range Attack: In this attack [114] (a.k.a., posterior corruption [32]), 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 only transaction-fee-based rewarding and infrequent or no check-points is denoted as a *stake-bleeding* attack [105]. *Countermeasures:* One mitigation is to lock the deposit for a longer time than the period of participation in the consensus [115]. The next mitigation technique is frequent periodic checkpointing, which causes the irreversibility of the blockchain with respect to the last checkpoint. Another option is to apply key-evolving cryptography [103] and forward-secure digital signatures [104], which require users to evolve their private keys, while already used keys are erased [113]. Hence, signatures cannot be forged in the case of compromise. The third mitigation technique is enforcing a chain density in a time-domain [105] for the protocols where the expected number of participants in each round is known (e.g., [21]). The last mitigation technique is context-sensitive transactions, which put the hash of a recent valid block into a transaction itself [105].

4) Side Effects of Countermeasures: Some countermeasures for threats in PoS protocols might impact the features of the blockchains. We note that secure multiparty coin-flipping protocol brings requirements on additional interactions among the consensus nodes, and thus it deteriorates the throughput of the protocol. On the other hand, the throughput does not

deteriorate when the leader and the committee are elected non-interactively by VRF.

VII. Replicated State Machine Layer

The Replicated State Machine (RSM) layer is responsible for the interpretation and execution of transactions that are already ordered by the consensus layer. 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 security threats of the RSM layer into two parts: standard transactions and smart contracts.

A. Transaction Protection

Transactions containing plain-text data are digitally signed by private keys of users, 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 the confidentiality of data [116]. Therefore, several blockchain-embedded mechanisms for the privacy of data and user identities were proposed in the literature, which we further elaborate on. Note that some privacy-preserving techniques can be applied also on the application layer of our stacked model but imposing higher programming overheads and costs (e.g., see Section IX-A, Section IX-F, and Section IX-G). 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 the privacy of transaction data and user identities in Figure 13.

Privacy Threats to User Identity: In many blockchains, user identities can be linked with their transactions by various deanonymization techniques, such as network flow analysis, address clustering, or transaction fingerprinting [116], [117], [118]. Moreover, blockchains designed with anonymity and privacy features (e.g., Zcash, Monero) are also vulnerable to a few attack strategies [119], [120]. *Countermeasures:* Various means are used for obfuscating user identities, including centralized [121], [122] and decentralized [123], [124], [125] mixing services, ring signatures [126], and non-interactive zero-knowledge proofs (NIZKs) [127], [128]. Some mixers enable internal linkability by involved parties [123] or linkability by the mixers [121], which are also potential threats. Unlinkability for all parties can be achieved by multiparty computation (MPC) [125], blinding signatures [122], or layered encryption [124]. Ring signatures provide unlinkability to users in a signing group [126], enabling only the verification of correctness of a signature, without revealing an identity of a signer.

Privacy of data: Blind signatures [129] and NIZKs such as zk-SNARKs [128] or (shorter) Bulletproofs [130] can be used for the preservation of data privacy. Another method is homomorphic encryption, 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 [131] utilizing zk-SNARKs, trusted hardware [132], and secure multiparty computations [133] embedded into these platforms. Privacy of data can be achieved even on blockchain platforms without embedded support of privacy-preserving constructs. For example, Zether [134] is built on top of the public smart contract platform Ethereum, and it provides a confidential payment mechanism that embeds the balance of users into (secret) exponents of ElGamal encryption. Other similar examples that deal with the privacy of data at the application layer of our stacked model are presented in Section IX-A, Section IX-G, Section IX-F.

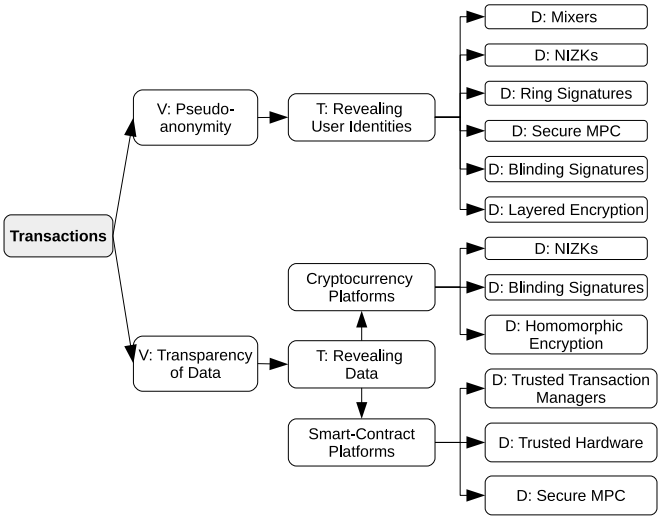


Fig. 13. Vulnerabilities, threats, and defenses of privacy threats (RSM layer).

2) Side Effects of Countermeasures: Since protocols of mixing services usually contain a few rounds (that may involve the creation of several transactions), all mixing services slow down the transaction throughput. The next blockchain feature that is influenced by some mixers is decentralization. As a consequence, centralized mixing services may misbehave and reveal linkable information of transactions, they can be DoS-ed, or they can steal the funds. Accountability for the misbehavior of centralized mixing service is provided in MixCoin [121] and Blindcoin [122] while the latter additionally provides internal unlinkability by a mixing service. In contrast to centralized mixers, decentralized mixers remove a trusted third party (i.e., no theft is possible) and provide stronger guarantees for unlinkability of transactions, e.g., CoinShuffle [124] and CoinParty [125] require at least two and $\frac{2}{3}m$ honest participants, respectively, to provide full unlinkability. Decentralization and availability are also impacted in solutions that utilize trusted hardware [131], [132].

The throughput of blockchains is also impacted in cryptographic countermeasures such as NIZKs, ring signatures, and blinding signatures. Ring signatures cause the large transaction size, which is linear with the number of participants in the anonymity set. The size of the ring signatures was optimized by cryptographic accumulators in [135], which in turn

enabled the improvement of the throughput. NIZKs utilized in ZeroCoin [127] produce large proofs as well. The proof size (and thus throughput) was further optimized by zk-SNARKs in ZeroCash [128]. However, the disadvantage of zk-SNARKs is the requirement for a trusted setup. This requirement is eliminated in Bulletproofs [130], which further decrease the size of proofs in transactions and thus improves on throughput.

B. Smart Contracts

Smart contracts introduced to automate legal contracts, 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 14.

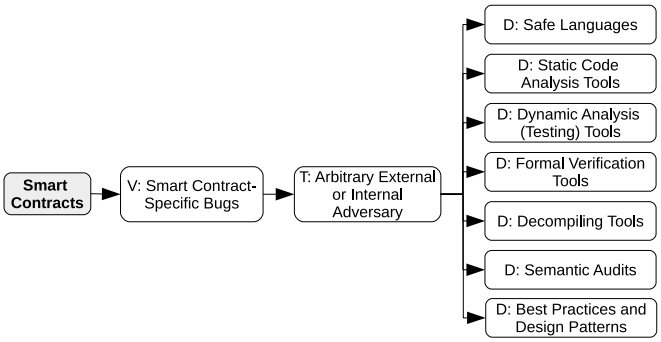


Fig. 14. Vulnerabilities, threats, and defenses of smart contract platforms (RSM layer).

Turing-Complete Languages: An important aspect of this smart contract language category is a large attack surface due to the possibility of 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*[8] 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 compilers [136], which opened the door for plenty of vulnerabilities. Hence, Serpent showed to be an unsuccessful attempt to simplify the coding phase. *Vyper*[9] is an experimental language designed to ease the audit of smart contracts and increase security – it contains strong typing, bounds checks, and overflows. *Yul*[10] 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* [137] is a type-safe language for Ethereum smart contracts. The major focus of this language 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)*[11] is a low-level language that is similar to Assembler. It aims to be simple and to support the creation of clean code; for example, 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 optimized use of the resources. *Solidity*[12] 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 [138], [139], [140], which resulted in many incidents in the past. Table IV of Appendix outlines the most prominent incidents and the associated vulnerabilities. In addition, the table classifies vulnerabilities according to an existing smart contract weakness classification (SWC) registry [141].[13]

Countermeasures: Mitigation techniques for such vulnerabilities are static or dynamic analysis (testing) tools, formal verification tools, security audits, as well as respecting best practices and using known design patterns [141], [142]. The literature contains various smart contract analysis tools for the detection of vulnerabilities [143], [144]. In the following, we give an overview of them:

- *Static analysis tools* such as linters, try to find vulnerabilities by inspecting the source code. For example, SmartCheck [145], Solhint,[14] Solium [146], and Slither[15] belong to this category. Another example, sCompile [147], works statically, but it also includes a dynamic component.
- *Dynamic analysis tools* seek vulnerabilities while executing the code of smart contracts. For example, simple forms of dynamic analysis are unit testing with handcrafted tests or replay testing [148], 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 [149], which generates unexpected, undefined, random, or invalid inputs to trigger a crash or reveal defects and vulnerabilities. Fuzzers, like ContractFuzzer [150], Echidna,[16] and Harvey [151] can be used for automated smart contract testing as well. Another technique of dynamic analysis is symbolic execution [152], 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

[8]https://github.com/ethereum/wiki/wiki/Serpent-%5BDEPRECATED%5D
[9]https://vyper.readthedocs.io/en/v0.1.0-beta.9/
[10]https://solidity.readthedocs.io/en/v0.5.10/yul.html
[11]https://lll-docs.readthedocs.io/
[12]https://solidity.readthedocs.io/

[13]Note that for some vulnerabilities there are no publicly available references on incidents supporting the existence of vulnerabilities.
[14]https://github.com/protofire/solhint
[15]https://github.com/crytic/slither
[16]https://github.com/crytic/echidna/

like Securify [153], Manticore [154], Oyente [155], and Osiris [156].

- *Formal verification tools* usually apply an abstract model or a semantic definition to check for the security and/or correctness properties of smart contracts. One type of this category is represented by semantic-based approaches that work with a semantic language specification defining the expected behavior of smart contracts. Examples of this type are FSolidM [157], and Kevm [158]. Other types of formal verification tools are semantic-based approaches that work with a behavioral model [159]. Several formal verification methods [160], [161], [162] 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 [163]). Additionally, such models can be applied to proving certain security properties [164].
- *Decompiling tools.* The source code of contracts is often not public in contrast to their bytecode. For this reason, bytecode decompilers, like Erays [165], Eveem,[17] or Porosity [166] can be used to (partially) reconstruct the source code of a contract. Additionally, there exist various static bytecode analyzers, like Maian [167], MadMax [168], Vandal [169], and automated exploit generators, like Teether [170] 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, which is achieved at the cost of limited expressiveness. Examples of this category are Pact, Scilla, Bitcoin Script, Ivy, and Simplicity. *Pact* [171] 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* [172] 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 in the last instruction of a contract, which simplifies proving safety and thus mitigates a few vulnerabilities. *Bitcoin Script* [173] 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* [174] 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.

[17]https://eveem.org/

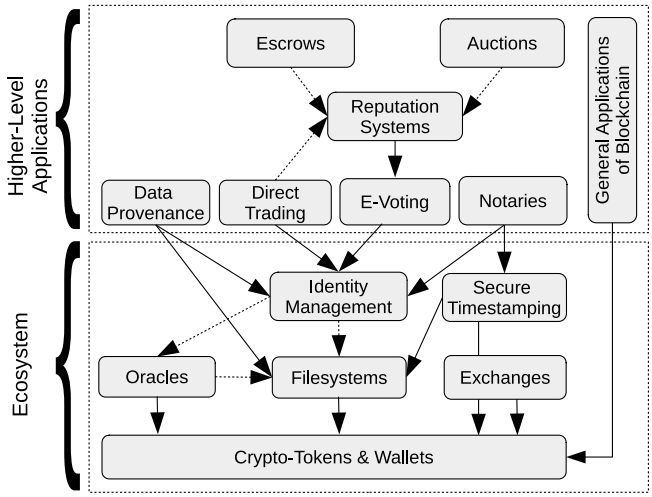


Fig. 15. Hierarchy in inheritance of security aspects across categories of the application layer. Dotted arrows represent application-specific and optional dependencies.

2) Side Effects of Countermeasures: Since all of the smart contract related countermeasures are performed before the deployment of smart contracts as part of the development stage of the blockchain-based applications, they do not negatively impact any blockchain features. Note that only countermeasures related to the operational stage of blockchain-based applications might influence blockchain features.

VIII. Application Layer: Ecosystem Applications

We present a functionality-oriented categorization of the applications running on or utilizing the blockchain in Figure 15, where we depict hierarchy in the inheritance of security aspects among particular categories. In this categorization, we divide the applications into categories according to the main functionality/goal that is to be achieved by using the blockchain. Security threats of this layer 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 denoted as an *ecosystem*, while we describe the rest of the applications in Section IX. 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**. We accompany the application layer with several incidents in Table III of Appendix.

A. Crypto-Tokens & Wallets

Besides blockchains that provide cryptocurrencies with native crypto-tokens, there are blockchain applications that use crypto-tokens to provide 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) [175]. All types of tokens require the protection

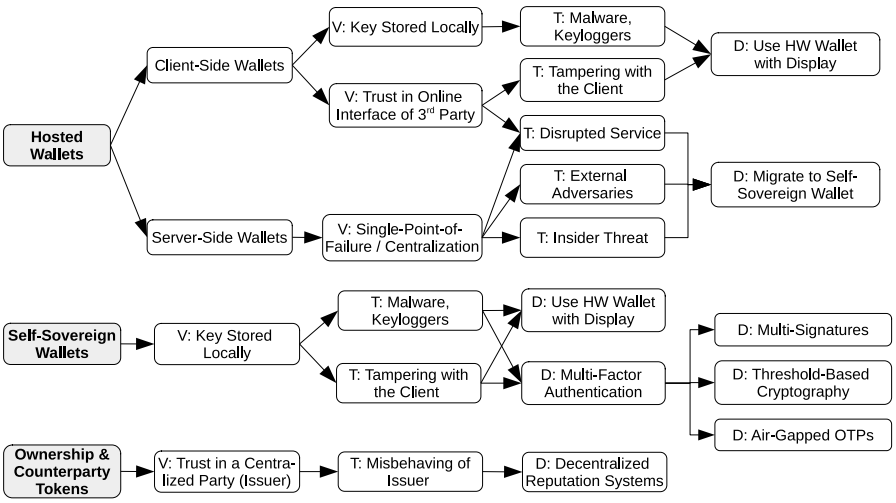


Fig. 16. Vulnerabilities, threats, and defenses of the crypto-token & wallets category (application layer).

of private keys and secrets linked with user accounts. For this purpose, two main categories of wallets have emerged: *self-sovereign wallets* (a.k.a., non-custodial) and *hosted wallets* [176], [177]. All crypto-tokens are exposed to technical and regulatory risks, while non-native tokens are also exposed to legal risks [175].

Self-Sovereign Wallets: Users of self-sovereign wallets locally store their private keys and directly interact with the blockchain platform using these keys, while they verify the inclusion of their transactions by SPV client software. 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,[18] MyEtherWallet[19]) as well as hardware wallets that store keys in sealed storage, while they expose only signing functionality (e.g., Trezor,[20] Ledger[21]). The next type of wallets enables functionality and security customization through a smart contract (e.g., TrezorMultisig2of3,[22] Ethereum MultiSigWallet,[23] SmartOTPs [178]).

Hosted Wallets: Hosted wallets require a centralized party, which provides an interface for interaction with the wallet and the blockchain. If a hosted wallet has full control over private keys, it is referred to as a *server-side wallet* (e.g., Coinbase[24]) while in the case when keys are stored in the user's browser, a wallet is referred to as a *client-side wallet* (e.g., Blockchain Wallet[25]). We refer the reader to works [176], [178] 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 16. Server-side wallets pose 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 [179], [180], [181], 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 thus 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 or stealing the key. 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).

In 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 [182], keyloggers [177], [183]). 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 (e.g., TrezorMultisig2of3, Ethereum MultiSigWallet), threshold-based cryptography [184], or air-gapped OTPs [178]. In the case of counter-party and ownership tokens presented 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 be built on top of them.

2) Side Effects and Implications of Countermeasures: A disadvantage of some multi-factor authentication solutions is that they require the execution of smart contracts, which increases the costs and might slow down the throughput of the system. In contrast, threshold-based cryptography constructs save these costs since they produce only a single signature,

[18]https://bitcoin.org/en/download
[19]https://www.myetherwallet.com/
[20]https://trezor.io/
[21]https://www.ledger.com/products/ledger-nano-s
[22]https://github.com/unchained-capital/ethereum-multisig
[23]https://github.com/ConsenSys/MultiSigWallet
[24]https://www.coinbase.com/
[25]https://blockchain.info/wallet/

which appears as it was made by a single party. However, threshold-based cryptography requires off-chain computation, in which the duration of execution is dependent on the number of co-signing parties.

Although self-sovereign wallets provide higher security in contrast to client-side and server-side hosted wallets, they impose overheads for storing the non-negligible part of the blockchain (i.e., headers) to validate the inclusion of signed transactions within their SPV client software – this is especially a concern when users have multiple SPV clients for multiple blockchains. BTC Relay is an early optimization attempt that reduces storage requirement by outsourcing the validation of transactions from the source blockchain to the target blockchain in exchange for a small fee paid to the relay nodes that submit headers to the target blockchain. However, the costs of BTC Relay were too high and its users tend to rather store headers of source blockchain on their own. Zk-relay [185] optimizes these costs by using zk-SNARKs with batched block validation, achieving an improvement by a factor of 187.

B. Exchanges

If the user wishes to exchange crypto-tokens, she might either directly find a counter-party wishing to exchange the opposite pair or approach an exchange that might be centralized or decentralized (DEX). In the case of centralized exchanges, the security threats and implications are due to centralization, and the only countermeasure is to use decentralized exchange solutions that we further focus on.

Direct Cross-Chain Exchange With Atomic Swap: Atomic swaps[26] assume two parties owning crypto-tokens in two different blockchains, and these parties wish to execute exchange atomically, i.e., either both of the parties receive the agreed amount or neither of them. The atomic swap protocol enables conditional redemption of the funds in the first blockchain upon revealing the hash pre-image (i.e., secret) that redeems the funds on the second blockchain. This protocol is based on two Hashed Time-Lock Contracts (HTLC) that are deployed by both parties in both blockchains, and it requires 4 transactions (see details in Appendix B).

Cross-chain DEX: Although atomic swaps are, in theory, sufficient means for the execution of fair cross-chain exchange, the situation is more complicated in practice. In particular, there might not exist a contra-party exchanging the opposite pair or the user might not be aware of it. This motivates DEXes, which facilitate the process of maintaining and matching the existing orders, act as a contra-party or intermediary, while guaranteeing fairness (e.g., Komodo[27]). The users match the orders, reward DEX, and afterward perform an atomic swap on their own. 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 different blockchains (e.g., $\mathbb{A}$ sells an asset for BTC and $\mathbb{B}$ buys it for ETH), DEX might serve as an intermediary who executes the three-way atomic swap [186] (see details in Appendix B).

Intra-Chain DEX: Some intra-chain DEX designs (e.g., Maker Market, EtherOpt, and Intrinsically Tradeable Tokens) require parties to post buy&sell offers on the blockchain, while smart contracts perform matches and execute trades. However, each placing of an order or its modification requires a payment for the inclusion of a transaction. 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. An example is 0x [187] protocol handling DEX of ERC20 tokens (e.g., applied in EtherDelta[28]). The next intra-chain exchange design is known as the *automated market maker* (AAM) [188]. AAM is applicable within a smart contract-based DEX that contains deposited reserves of traded ERC20 tokens; examples are Euler [189], Bancor [190], and Uniswap.[29] AAM provides high liquidity since users are not required to match their orders, and they can directly do the exchange with the smart contract.

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.* [191].

1) Security Threats and Mitigations: We present an overview of vulnerabilities, threats, and defenses related to the exchanges category in Figure 17.

In **centralized exchanges**, threats are caused by external and internal adversaries, and they are identical to those of server-side hosted wallets (see Section VIII-A) since server-side wallets always provide exchange services. So far centralized exchanges posed the most attractive target for adversaries that have caused huge financial losses [192]. There are many operation security (OPSEC) countermeasures such as multi-factor authentication, split of the funds to hot and cold wallets, however, none of them eliminates the single-point-of-failure coming from the centralization. Therefore, effective mitigation from the user point of view is to use decentralized exchange solutions such as DEXes and atomic swaps; however, they also contain some vulnerabilities.

Different blockchains of **cross-chain** decentralized (direct and DEX-based) exchanges might have a different time to finality, and thus 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, 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

[26] https://en.bitcoinwiki.org/wiki/Atomic_Swap
[27] https://komodoplatform.com/atomic-swap-technology/
[28] https://etherdelta.com/
[29] https://uniswap.io/

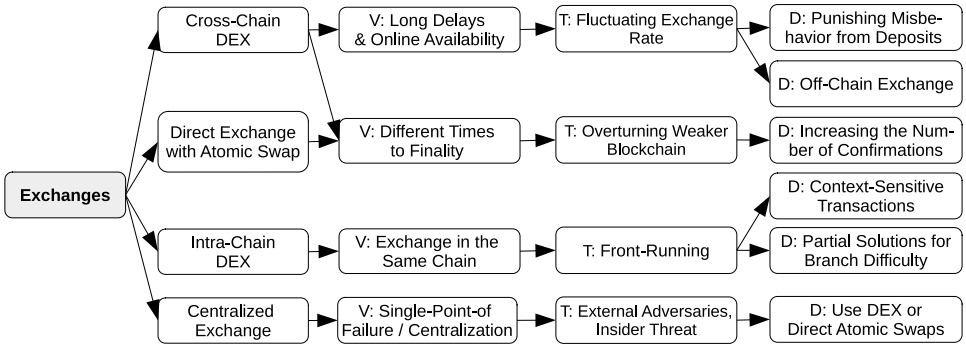


Fig. 17. Vulnerabilities, threats, and defenses of the exchanges category (application layer).

with the use of TEE [193] (see below). 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.

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 transactions containing higher fees and by optimizing network latency [92]. Moreover, such adversaries might compete with each other by "bidding" a higher transaction fee [92], 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 to these threats is represented by context-sensitive transactions [105], which do not allow overturning of the blockchain, only its extension. The same effect can be achieved by partial solutions included in branch difficulty computation [18], [25], [86], [90]. Note that context-sensitive transactions and partial solutions are a means of the consensus layer.

2) Side Effects and Implications of Countermeasures: We assume centralized exchanges as the baseline that requires only two transactions for settlement of cross-chain exchange or one transaction for intra-chain exchange. Note that such settlement is done only sporadically since centralized exchanges inherently work off-chain in real-time. In contrast, direct atomic swaps require 4 transactions while three-way atomic swaps require 6 transactions. Therefore, using these constructs negatively affects the throughput of blockchains. Nevertheless, the performance can be improved by off-chain execution of atomic swaps, which provides almost immediate response, e.g., off-chain swaps are possible in two-parties payment channels as well as their extension to multiple parties known as the lightning network [194]. Such off-chain solutions greatly improve the scalability, since parties can transact directly and involve blockchain consensus only when they wish to settle their balances (e.g., once per day or week). However, to avoid misbehavior in which a "stale" balance is settled on-chain, these systems require that the parties constantly monitor the blockchain state. Such an always-online assumption can be relaxed by employing watching services [195], [196], [197] (a.k.a., watchtowers), which, however, incur extra costs.

TEE can be also utilized as an off-chain means that improves the throughput of exchange service. For example, Tesseract [193] is a real-time exchange that leverages TEE for communication with users and a target blockchain. To enter the system, users submit time-locked refill transactions paying to Tesseract's controlled address in the target blockchain, and then Tesseract uses its SPV client to verify their inclusion. Existing users submit bid&sell requests to Tesseract, which performs matching and executes trades within TEE. When users want to sync with the on-chain state, they ask Tesseract to generate settlement transactions. Since decentralization would be impacted by using a single service of Tesseract, which might censor user requests, the authors incorporate the Paxos consensus protocol among multiple mutually untrusted Tesseract nodes; this also increases the fault-tolerance and avoid funds of users becoming stuck in contrast to one instance of Tesseract. We note that censorship evidence (not resistance) could be alternatively ensured in Tesseract by smart contract-based censorship resolution [198], which, however, implies some extra costs for smart contract execution. Finally, TEE-based solutions might be vulnerable to attacks on trusted hardware. As a mitigation technique to reliance on a trusted manufacturer of TEE, it is possible to use a quorum of several redundant TEEs from multiple manufacturers. Furthermore, it is important to note that the assumption about the code executed in TEE is its bug-freeness, and thus one might not use return-oriented programming or other techniques to ex-filtrate sealed secrets or private parameters; this is an out-of-the-scope attacker model for TEE.

C. Oracles

Oracles (a.k.a., *data feeds*) are trusted entities that provide plausible data that reflects the state of the world beyond the blockchain. The authors of [199] and [200] define a few security properties of oracles in smart contract platforms:

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

Integrity: Provided data 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 data. Therefore, an oracle should support such parameters and their handling.

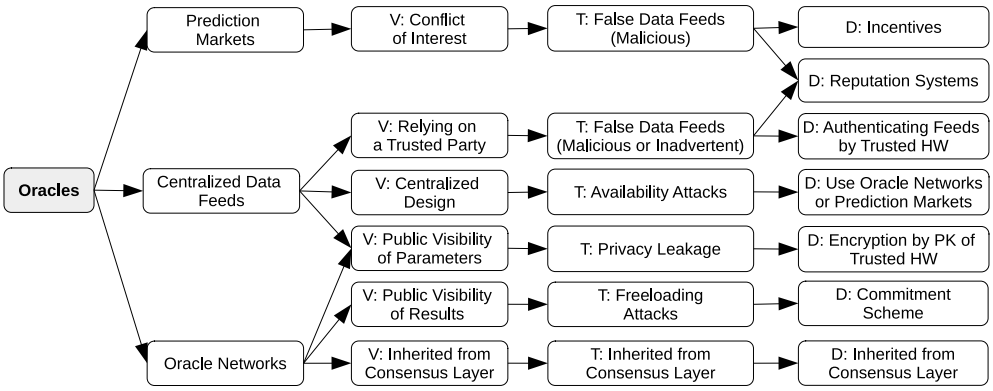


Fig. 18. Vulnerabilities, threats, and defenses of the oracles category (application layer).

Availability: Since the execution of dependent smart contracts relies on data feeds delivered by oracles, they need to provide high availability.

We categorize existing oracles according to security implications into three categories. *Prediction markets* (e.g., Augur [201], Gnosis[30]) were created for the purpose of trading the outcome of events – individuals are incentivized to accurately wager on outcomes serving as data feeds, while outcomes are provided by either a centralized reporter or a quorum of reporters. *Centralized data feeds* provide arbitrary data from a single centralized source, and they build on existing blockchain platforms (e.g., Oraclize,[31] Town Crier [202], PDFS [199]). Finally, *oracle networks* internally run a consensus protocol for decentralized agreement on data (e.g., ChainLink [200], Witnet [203]).

Prediction Markets: Augur [201] is a solution designed as the Ethereum smart contract, and it uses its own reputation token. The user creating the market specifies a designated reporter, who delivers the result of the event after it happens. However, the reporter might not report the result or report incorrect results. When the reporter does not report within the specified time frame, Augur shifts the role of a designated reporter on a first-come-first-serve basis. After reporting the outcome, the Augur users have a specific time frame to run a decentralized dispute resolution, and thus obtain a different outcome of the event in the case of misreporting. Augur incentivizes its users to report correct outcomes and file disputes only in justified cases by rewards and deposit-based bonds. Another example of prediction markets is Apollo from Gnosis, which had originally a centralized data source facilitated by Gnosis but was replaced by a decentralized one in version 2.0.

Centralized Data Feeds: Oraclize enriches the data provided to smart contracts by authenticity proofs that are built upon various technologies such as auditable virtual machines and trusted computing. Since authenticity proofs can be large, Oraclize can store these proofs in the distributed file system IPFS[32] instead of directly providing them to the smart contracts. Town Crier [202] is an approach that provides authenticated data feeds to smart contracts by bridging them with public webs through a TEE component. A linkage of TEE with a smart contract is made by storing a public key (PK) of TEE at the smart contract of Town Crier. It relies on the X.509 public key infrastructure (PKI), due to which, the provided data are provably authenticated. PDFS [199] allows content providers to link their Web resources with corresponding smart contracts in the blockchain. In PDFS, the data of content providers are managed 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 that updates are consistent with the past, the authors apply a history tree data structure [204]. The authors additionally support the means to publicly prove censorship by a content provider.

Oracle Networks: ChainLink [200] 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 utilizes the BFT protocol to reach a consensus on a final aggregated value. Witnet [203] is 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 in parallel, forming a DAG.

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the oracle category in Figure 18. In the following, we describe these threats as well as possible defense techniques.

Prediction Markets: May suffer from conflict-of-interest since the creator of the market specifies a data reporter, who might also participate in the market and later report false data for her convenience. Since a prediction market might

[30]https://gnosis-pm-contracts.readthedocs.io/en/latest/
[31]https://www.oraclize.it/
[32]https://ipfs.io/

yield significant financial value for the users with a "correct" guess, the malicious reporter might bribe other reporters of dispute resolution round and still be profitable. Therefore, the incentive protocols of prediction markets must count on this situation and incorporate feasible rewards for honest reporters. Another mitigation for similar attacks is to keep reporters accountable and maintain their reputation in a decentralized fashion (Section IX-B), which involves identity management and verification (see Section VIII-E).

Centralized Data Feeds: Rely on a trusted party [199], [205] that may misbehave or accidentally produce wrong data. For both cases, decentralized identity management can bring accountability while reputation systems further build on it, which disincentivizes malicious behaviors. Another option to cope with possible misbehavior of a trusted party of oracle service is to embed the logic of oracle service into a TEE component [202], whose code is publicly attested. TEE component can interact with the external world using X.509 public key infrastructure (PKI), due to which obtained data are provably authenticated. Since some requests of data feeds might contain private parameters,[33] they can be encrypted by a PK of the TEE and further processed within TEE that communicates with its remote data provider through an encrypted channel, while communication is facilitated by oracle's service operator, as demonstrated in Town Crier [202]. Centralized data feeds might be subject to attacks on availability, leading to interrupted service. A mitigation technique is to use solutions with a higher redundancy, such as oracle networks.

Oracle Networks: Eliminate trust in a single party by running a consensus protocol either natively [203] or utilizing an existing smart contract platform to facilitate the service and its consensus [200]. Running the native consensus protocol of an oracle network imposes the security threats related to the consensus layer (see Section VI). Specific threats to oracle networks are freeloading attacks, in which an oracle provider might copy a publicly visible value provided by other oracles without any effort. The authors of ChainLink [200] propose the usage of a commitment scheme to cope with this attack.

2) Side Effects and Implications of Countermeasures: The data provision time of prediction markets may be too long for many applications, and they are convenient to use only for specific use cases that are limited to provided data events. The data provision time is prolonged even more in the case of disputes, whose resolution may require several days or weeks. In contrast to limited data of prediction markets, centralized data feeds enrich the data domain and significantly shorten the provisioning time.

In the case of oracle providers that offer authenticated data feeds using trusted hardware [202], [205], a vulnerability in trusted hardware (caused by a manufacturer) may result in the entire data feed being compromised. As a mitigation technique to reliance on a trusted manufacturer of TEE, it is possible to use a quorum of several redundant TEEs from multiple manufacturers.

Since ChainLink [200] is an oracle network running over the public smart contract platform, it imposes significant execution costs. 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. When freeloading attacks are resolved by a commitment scheme, it negatively impacts the provisioning delay and costs for data providers, who have to submit another transaction with a commitment.

D. Filesystems

Filesystems (FS) serve as a distributed data storage infrastructure that borrows ideas from peer-to-peer storage systems, while additionally incentivizing data preservation by tokens.

1) Fully Replicated FS With Ledger: A naive approach is to store the full content of data at the blockchain, and thus achieve full data replication, extremely high *data durability* (i.e., availability of data) as well as *network expansion factor* (i.e., storage overhead). An example is storing data using the instruction *OP_RETURN* in Bitcoin or storing data as key:value pairs within Namecoin.[34] However, such an approach results in high storage overheads required for full replication of the data among the consensus nodes.

2) Partially Replicated FS With Ledger: To decrease the costs while preserving reasonable durability, partial replication of the data with erasure encoding is often used (e.g., Permacoin [78], Storj [206], and KopperCoin [207]). 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 minimum number of shares required for data recovery. Permacoin incorporates Proof-of-Retrievability in the consensus layer, where consensus nodes store large segments of data provided by an authoritative file dealer. KopperCoin follows a similar approach, but it does not need the trusted dealer for the initial distribution of data files since files are uploaded by the users. Storj uses a 3rd party distributed ledger for storage of metadata, such as file hash, network locations of copies, and Merkle roots of data. Permacoin, KopperCoin, and Storj enable probabilistic audits using Proof-of-Retrievability, which proves that a node stores certain data at the time of the challenge. Filecoin is an incentive mechanism of any distributed FS (e.g., IPFS), and in contrast to previous works, it can guarantee the data possession over a certain time range in a setting of Proof-of-Spacetime. Moreover, Filecoin uses Proof-of-Replication [208], which guarantees physically unique copies of data for each node.

3) Partially Replicated FS Without Ledger: IPFS and Swarm[35] utilize the concept of distributed hash tables (DHT). 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., its hash). Since the lookup service and data storage are (partially) distributed, a change in the set of participants causes only a negligible disruption of availability. IPFS does not contain any incentive mechanism and the availability of the data is dependent on its popularity. Although IPFS does

[33] All transactions of permissionless blockchains are visible to public.

[34] https://bit.namecoin.org/

[35] https://swarm-guide.readthedocs.io/en/latest/

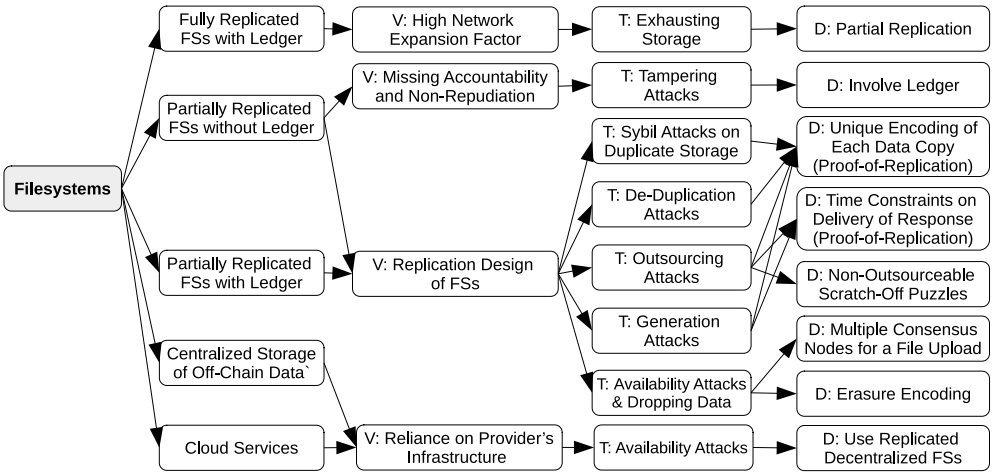


Fig. 19. Vulnerabilities, threats, and defenses of the filesystems category (application layer).

not involve a blockchain, it may achieve its properties. In particular, nodes may optionally store a BitSwap ledger that logs data transfers with other nodes.

4) Centralized Storage of Off-Chain Data: 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 data, cloud services are promising approaches for decentralized yet manageable data storage, for which integrity and consistency proofs are stored on some blockchain.

a) Security threats and mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the filesystems category in Figure 19. While fully replicated and partially replicated decentralized filesystems handle **availability** using decentralized infrastructure, centralized storage with integrity proofs and cloud services relies on a centralized provider. In a **Sybil attack** on a replicated FS, a malicious node claims the 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 [208]. In an **outsourcing attack**, a malicious consensus node claims the 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 storing of a huge volume of generated data.

On top of the 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 [208]. 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. Similar mitigation for outsourcing attacks is the use of non-outsourceable scratch-off puzzles [60], [78], in which the computation of the puzzle requires access to the storage in random order; hence, 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.

b) Side effects and implications of countermeasures: Although unique encoding of each data copy thwarts several attacks, on the other hand, it imposes higher overheads for file distribution on clients, which might negatively influence the throughput of data. Similarly, additional overheads on the client during the file upload is imposed by the use of multiple consensus nodes for file upload. Although erasure encoding aims to decrease costs, it must be viewed in a trade-off with the availability of data that is negatively affected by it.

E. Identity Management

Identity management refers to binding identities of entities to their public keys. This concept is also referred to as Public Key Infrastructure (PKI), and it has a few security goals [209]:

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.

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 [210]. However, this situation has been changed with the invention of blockchains; in particular, their immutability feature (Appendix A1).

Namecoin[36] is a native blockchain that facilitates identity management since it allows for the unique registration of key:value mappings. However, searching for a value associated with a key requires full storage and traversal of the blockchain, which is costly. Blockstack [211] is a similar approach as Namecoin, providing decentralized DNS, but in contrast to Namecoin, it off-chains the data storage of domain name mappings and keeps only references to hashes of zone-files in its blockchain. Zone-files (and their referred DNS entries) are stored off-chain. Certcoin [209] 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, imposing only a logarithmic time complexity in the number of registered users (in contrast to linear time complexity of Namecoin). Furthermore, to speed up the PK lookup queries, Certcoin leverages the concept of DHT (see Section VIII-D), which enables it to achieve a constant lookup time complexity. Ethereum Name Service[37] (ENS) maps human-readable domain names to Ethereum addresses in a similar fashion as in DNS. The root domain of ENS is maintained by a multi-signature smart contract owned by trustworthy individuals from the Ethereum community. Similarly, uPort [212] utilizes smart contracts to keep a registry that maintains a mapping of the user addresses to hashes of claims[38] 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,[39] in which, identity is associated with several keys serving various purposes. ShoCard [213] is another example that builds on top of existing public blockchains, but in contrast to the previous examples, it builds a sidechain containing encrypted identity-specific data such as biometric data, scans of IDs, etc. The user may then decide to whom she will reveal the encrypted data. The Sovrin [214] is an example providing a public permissioned blockchain that consists of consensus nodes approved by Sovrin. It focuses on high throughput and low operational costs. Decentralized Identifiers (DIDs) [215] represent a new type of universally unique identifiers whose control is decentralized since all roots of trust are contained in the blockchain and each entity might create its own root of trust. DID employs the same hierarchical scheme for globally unique strings as URI, and it maps strings to DID documents containing data such as PKs, endpoints of the entity, or links to off-chain data. Only the owner can create, manage, and prove ownership of her DID entries.

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the identity management category in Figure 20. As mentioned by Kalodner *et al.* [216], most of the solutions address the problem of mapping names to values but for identity management, 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, a human arbitration or a trusted party is needed. For this purpose, oracles (see Section VIII-C) might deliver verified data about the identity of users.

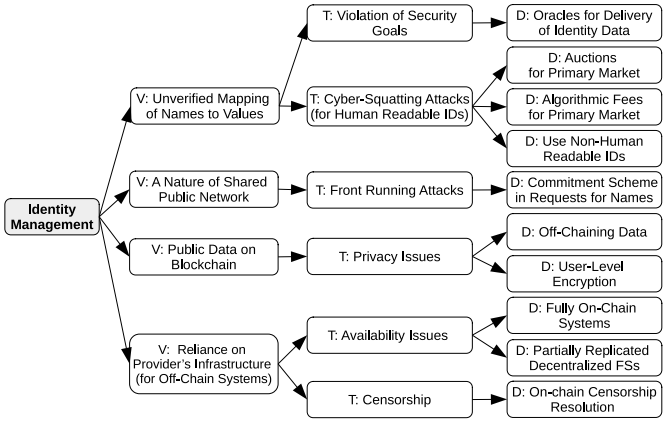


Fig. 20. Vulnerabilities, threats, and defenses of the identity management category (application layer).

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 to cyber-squatting attacks, in which anybody might seize an ID that does not belong to her and then sell the ID in the secondary market at an inflated price. Kalodner *et al.* [216] 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. These two strategies stand for *auctions* and *algorithmic fees*, both having their respective cons. Auctions are problematic since they may be initiated at any time, and some potentially interested bidders might not be available. An improvement is to specify a fixed time when auctions start. Another approach for coping with cyber-squatting is an algorithmic fee solution, which assigns the price based on the deterministic observables, such as length of the name, a rank of the domain, occurrence of human-readable words, etc. Nevertheless, this approach may require a data feed provider (see Section VIII-C). In contrast to human-readable identifiers, non-human readable identifiers, such as DIDs, do not suffer from the cyber-squatting 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 name 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, while in the second step she submits the plain text of the name.

Further, identity-related user data that are published on the blockchain are subject to certain privacy issues. Mitigation is to keep only integrity information (such as hashes) on

[36]https://www.namecoin.org/
[37]https://ens.domains/
[38]Claims as such belong to the notaries category (see Section IX-D).
[39]https://erc725alliance.org/

the blockchain, while data should be stored off-chain [211], [212], [213] or encrypted by the user's private key [213]. Moreover, all the approaches that rely on off-chain storage and service provisioning are vulnerable to availability issues (e.g., [212], [215]) and censorship attacks (e.g., [214]). Mitigation that provides censorship evidence is an on-chain censorship resolution [198].

2) Side Effects and Implications of Countermeasures: Side effects of oracles depend on their category, and we mention them in Section VIII-C. If a beginning time of the auctions on the primary market is fixed, bidders can be DoS-ed and prevented from bidding. Anonymization networks and VPNs used by bidders might mitigate this problem. While off-chaining of identity-related data helps to cope with privacy issues and decreases operational costs, on the other hand, it means dependency on centralized storage, which negatively impacts the availability of data. Possible mitigations are partially replicated decentralized filesystems (see Section VIII-D).

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 simple examples of secure timestamping are CommitCoin [217], STAMPD,[40] Bitcoin.com Notary,[41] and OriginStamp [218], all enabling to post a document's hash into a single blockchain transaction. OpenTimestamps[42] and POEX.IO[43] 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.

1) Security Threats and Mitigations: Figure 21 depicts a taxonomy of vulnerabilities, threats, and defenses related to the secure timestamping systems. Since these systems have a narrow principle of operation and provided functionality, their attack surface is very limited, too. The main security threats stem from inaccuracy and imprecision of timestamps provided by blockchain network as well as aggregation delays of certain secure timestamping services. As a consequence, the result of certain disputes might be influenced. A possible mitigation technique to improve the accuracy of timestamps is to use timestamping authorities [65] or partial solutions for block timestamp computation [18].[44] A mitigation technique for long aggregation delays is to employ timestamping authorities or use one transaction per hash of the timestamped record. Another class of attacks concerns the availability of timestamped data, for which decentralized filesystems might

[40]https://stampd.io/
[41]https://notary.bitcoin.com/
[42]https://opentimestamps.org/
[43]https://poex.io/
[44]Note that this is a countermeasure specific to the consensus layer.

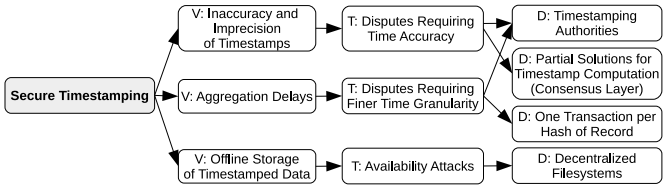


Fig. 21. Vulnerabilities, threats, and defenses of the secure timestamping category (application layer).

be utilized as a mitigation technique while storing data in encrypted or plaintext form (depending on the use case).

2) Side Effects and Implications of Countermeasures: Although one transaction per hash of a timestamped record mitigates the impact of aggregation delays in solutions such as OpenTimestamps and POEX.IO, on the other hand, it requires a higher amount of data posted to blockchains, and thus it deteriorates the throughput and imposes higher costs. Therefore, the choice of either an aggregated solution or a single hash per record depends on a particular use case.

IX. Application Layer: Higher-Level Applications

In this Section, we elaborate on more specific higher-level applications as opposed to ecosystem applications. In detail, we deal with the following categories: (1) **e-voting**, (2) **reputation systems**, (3) **data provenance**, (4) **notaries**, (5) **direct trading**, (6) **escrows**, (7) **auctions**, and (8) **general application of blockchains**. We describe each of the categories, present a few examples, and then summarize the potential security vulnerabilities, threats, and defenses. For other detailed reviews of blockchain applications, we refer the reader to Casino *et al.* [11] and Zheng *et al.* [12], which in contrast to our work follow domain-oriented classification.

A. E-Voting

Kiayias and Yung *et al.* [219] and Groth [220] state several properties that are desirable in e-voting applications:

Perfect Ballot Secrecy: Implies that finding partial results (i.e., partial tally) before the voting finishes is possible only if all voters are involved in its computation.

Fairness: The final tally can be computed only when all participants already had a chance to cast their vote.

Public Verifiability: Any public observer can verify the validity of all votes and final tally. This is achieved by using a public bulletin board (e.g., blockchain). A consequence of the public verifiability is *dispute-freeness*, i.e., the result of the voting is indisputable.

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 is to prevent vote-selling and post-election coercion.

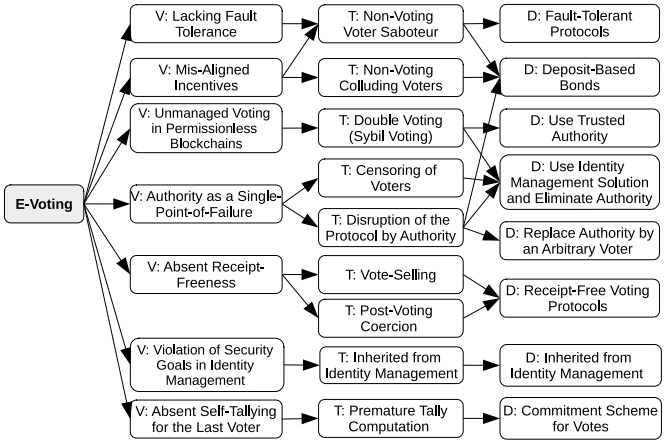


Fig. 22. Vulnerabilities, threats, and defenses of the e-voting category (application layer).

E-voting [221], [222] has tried to mimic many of the security properties provided by paper voting. In decentralized e-voting, the protocol is carried out in phases and requires a multiparty computation (MPC) [219], [220] executed by the voters. Decentralized voting involves an interaction among participants and is less robust concerning 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, public verifiability, enforcing protocol rules by the smart contract, and higher availability [223].

An example of a decentralized e-voting is the Open Voting Network (OVN) [224], which for the first time utilizes blockchain as an instantiation of the public bulletin board. OVN is implemented using Ethereum smart contracts, and it enables boardroom voting of up to ∼50 voters with support for two 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 outcome of voting or compromise the privacy of voters (i.e., cast votes). Although OVN preserves the privacy of voters and provides self-tallying, it does not provide receipt-freeness. OVN uses deposit-based penalties to incentivize the authority and voters to actively participate. Zhang *et al.* [225] present a distributed e-voting scheme that uses the blockchain, but the proposed protocol (like OVN) does not provide a fault tolerance – a restart of voting is required if any voter does not cast her vote. This enables sabotaging of the voting process by a single malicious voter. Venugopalan *et al.* [226] propose a boardroom voting over Ethereum, which supports n voting choices and provides a fault tolerance mechanism. Li *et al.* [227] propose an approach that assumes an interactive honest verifier for the zero-knowledge proof presented; however, the verifier can select a biased challenge, which enables collusion of the verifier and the authority.

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the decentralized e-voting category in Figure 22. The first e-voting-specific group of threats represents 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. Mitigation is to use receipt-free voting protocols that thwart both attacks [228], [229]. Existing solutions for achieving receipt-freeness assume a secret channel (bi-directional [228] or uni-directional [230]), use deniable encryption [231], [232], or employ randomizers [229].

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 permits voters to vote.[45] Another issue is a possibility of voters not voting despite enrollment, resulting in the sabotage of the voting round (possible in [220], [224], [225]) or privacy issue related to more fine-grained inference of the actual votes of the remaining voters who voted. Deposit-based bonds might be employed as penalties for saboteurs and disincentivize such behaviors [224]. Another countermeasure for saboteurs is a fault-tolerant voting protocol (e.g., [226], [233]), in which the remaining honest voters can recover the final tally even without votes of saboteurs. Since e-voting might assume verified identities of all participants, the next group of threats that is worth noting 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, as the last participant can compute the tally and only after that decide on her vote. 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 [224].

2) Side Effects and Implications of Countermeasures: Receipt-free voting protocols can protect against vote-selling or coercion but on the other hand, they imply additional computational overhead for voters, which increases the cost of running a decentralized protocol. Although authority can prevent double voting, it is trusted with managing voters honestly and completing all actions required for progressing the protocol from one stage to the next stage (e.g., [224], [226]). 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 voter for the purpose of execution of the voting protocol (but not for managing voters). Regarding the management of voters, an important threat is the possibility that the authority might censor some eligible voters. A solution for eliminating the authority (or a delegation of this problem to a different application type) is permitting voters to vote upon successful registration of their identities within a certain identity management application (see Section VIII-E). Fault-tolerant voting protocols impose additional overheads, where all remaining voters must actively participate in the recovery round – this implies additional costs on such voters, and it slows down the protocol. Similarly, a commitment scheme for coping with violation of self-tallying

[45]This is represented by know your client (KYC) compliance, and it is related to the principles of permissioned blockchains.

in the case of the last voter implies additional overheads and costs for a dedicated commitment 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 the voting of parties/users that independently analyze the history of interactions/records produced by entities in a reputation query.

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

1) Rating by Arbitrary Participants: A privacy-preserving reputation system for e-commerce was proposed in [234]. The authors utilize blinding signatures and merchant-issued tokens to achieve the privacy of reviews and avoid bad-mouthing and ballot-stuffing attacks. A feedback-based reputation approach that utilizes the incentive-based scheme of the Bitcoin network is proposed in [235]. In this approach, any consumer might rate the service of the producer, while obtaining a voucher for the feedback. Zhao *et al.* [236] propose a reputation management scheme that utilizes additive secret sharing to achieve the privacy of participants in reputation assessments.

2) Rating by Several Selected Participants: An example of such a reputation-based application is related to accreditation of educational institutions by other higher-level institutions and organizations [237].

3) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the reputation systems category in Figure 23. Specific security threats to reputation systems with an arbitrary number of legitimate participants are **bad-mouthing**, **ballot-stuffing**, and **whitewashing** attacks [234]. In bad-mouthing attacks, the customer (e.g., competitor) lies about the product or service, while in the ballot-stuffing attacks, the service provider might increase her reputation by herself. Bad-mouthing can be mitigated by filtering only authorized participants to submit reputation assessments (e.g., by review tokens [234], [235]). Although bad-mouthing cannot be completely prevented, it requires participants to spend resources (e.g., buying a product or paying transaction fees) 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 – the service provider creates a new service with a neutral reputation, which is unlinkable to her previous service. To mitigate this attack, oracles

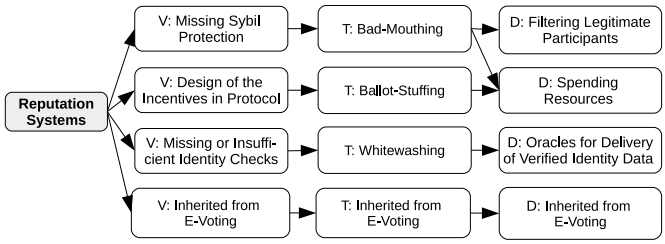


Fig. 23. Vulnerabilities, threats, and defenses of the reputation systems category (application layer).

for obtaining verified data about identities of entities can be employed, possibly as part of the identity management system.

Since reputation systems resemble e-voting in general, concerning security threats are inherited from there (see Section IX-A) and its dependency on identity management (Section VIII-E). In particular, only authorized participants are allowed 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.

4) Side Effects of Countermeasures: Since some countermeasures in reputation systems are inherited from the e-voting and oracles categories, the side effects are also inherited from these categories.

C. Data Provenance

Data provenance represents the ownership history of an arbitrary object. However, in the cyber-world, objects are represented by data that can be changed, and thus the history must account also for the modifications [238]. Data provenance with the use of blockchains has the potential to resolve various disputes and issues related to intellectual property, authorship, the validity of certificates or other issued documents, etc. Data provenance assumes known verified identities of the involved parties.

An application of data provenance is supply-chain management [239], [240], where the goal is to resolve potential issues in the traceability of goods and provenance of associated data [241]. Blockcloud is an approach that utilizes blockchains for data provenance in cloud computing [242]. 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 [243]. ChainAnchor [244] is a framework for the commissioning and decommissioning of IoT devices in a cloud-based ecosystem. In a commissioning procedure, devices prove their manufacturing provenance to a verifier in a privacy-preserving fashion without a need for interaction with the manufacturer. An additional goal of ChainAnchor is to reward owners of IoT devices for sharing data in a privacy-preserving manner. A data provenance approach that focuses on the integrity of IoT-generated data is proposed in [245]. A framework to achieve data provenance of multimedia objects such as artworks and books was proposed in [246]. The authors use watermarking techniques to embed transaction metadata of objects into the objects themselves to prove data tampering.

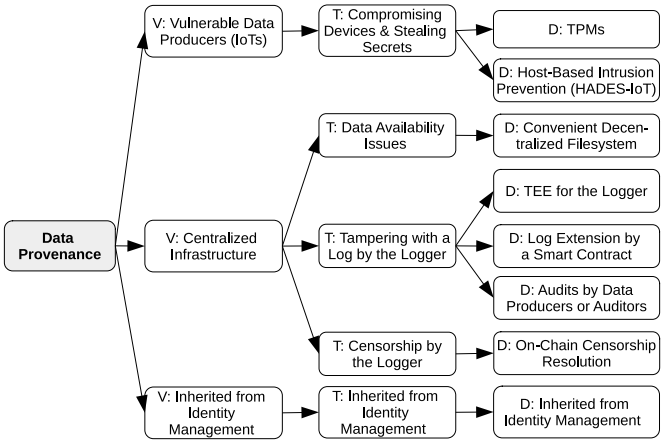


Fig. 24. Vulnerabilities, threats, and defenses of the data provenance category (application layer).

Catena [247] guarantees a non-equivocation of its append-only log and relatively low storage overheads (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 Catena transaction. The same method of binding consecutive transactions in an append-only log was utilized in Contour [248]. In contrast to general Catena, Contour focuses on non-equivocation during the distribution of open-source software packages by a specified authority. Grech and Camilleri [237] elaborate on the usage of blockchain for the issuance of educational certificates by academic institutions. Such certificates might be issued by institutions whose identities are verified (see Section VIII-E).

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the data provenance category in Figure 24. Since data provenance infrastructure might involve simple IoT devices and sensors that are often not updated nor physically protected, it is important to ensure that these devices are tamper-proof and no secret can be stolen from them. As a countermeasure, Trusted Platform Modules (TPMs) may be used if they are available. We note that the current trend is to involve TPMs in the hardware of contemporary IoT devices. However, TPMs are often not available in numerous legacy IoT devices. In such cases, it is possible to leverage kernel-space and user-space memory isolation as part of the intrusion prevention system (e.g., [249]). Another vulnerability originates from a possible centralized logger component of data provenance solutions (e.g., [199], [247], [248], [250]). Hence, availability issues for data storage must be considered and possibly resolved by a convenient decentralized filesystem approach (see Section VIII-D). Another threat concerning the centralized infrastructure of loggers is the possibility of data tampering and censorship. Data tampering can be detected by data producers or auditors that do periodic audits [251]. To cope with censorship, an on-chain smart contract-based censorship resolution can be utilized [198], [250].

2) Side Effects and Implications of Countermeasures: Although auditors might detect tampering with data by a logger entity, their periodic activity implies high operational costs. A possible option to save such costs is an auditor-free update of the log using a smart contract [199], alternatively combined with TEE [250] to enforce even stronger properties (i.e., the correctness of the execution by logger). However, TEE has also its cons, which we discussed above. Moreover, since countermeasures of the data provenance category depend on the identity management and filesystems category, the side effects are inherited from them as well.

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 [252]; hence, notary systems assume known verified identities of involved parties who do the vetting. In addition to the above two functionalities, the definition of the 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 documents in a database of a notary service provider (e.g., PADVA [198]) or whether to keep it privately at the client-side (e.g., Blockusign[46]).

A blockchain-based notarization platform on Ethereum was proposed in the post [253], where an arbitrary number of users/entities with verified identities may sign/approve the documents and their new versions, respectively. The proposal assumes a certification authority that verifies the identities of involved entities, and as an example, the authors suggest the use of the ERC 725 standard [254]. ADVOCATE [255] is an approach for notarization of agreements about personal data processing in IoT between owners of IoT devices and data processing services – both must co-sign an agreement. Mizrahi [256] proposes a system for property ownership, where, all ownership transfers can be executed without any trusted party, but the trusted party is required for introducing the initial ownership record to the blockchain. The ownership register for vehicles was proposed by Notheisen *et al.* [257], where trusted third parties, such as police departments and transport authorities provide and verify vehicle-specific information. SilentNotary[47] is a smart contract-based system for self-certifying of files produced by registered users. PADVA [198] is a Transport Layer Security (TLS) notary service realized as a smart contract-based two-party agreement (i.e., Service Level Agreement). PADVA introduces notary entities that are obligated to periodically check the validity of PKs in a specified set of certificates.

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the notaries category in Figure 25. Notaries inherit security threats related to timestamp accuracy from the secure timestamping category (see Section VIII-F). In addition, since they assume verified identities of involved parties, they inherit security issues from the identity management category (see Section VIII-E). In particular, many notary services assume a centralized identity management system, which might be subject to tampering or censorship issues. Next, a very specific

[46]https://blockusign.co/
[47]https://silentnotary.com/

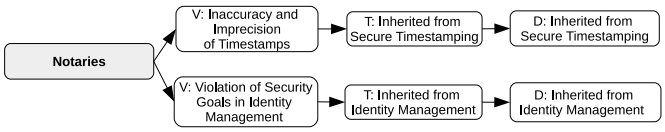


Fig. 25. Vulnerabilities, threats, and defenses of the notaries category (application layer).

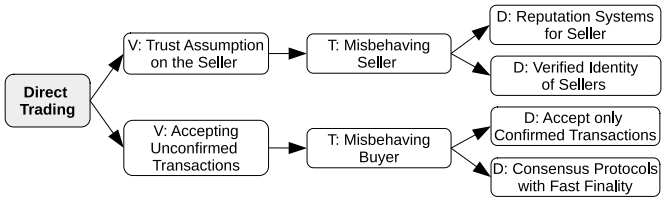


Fig. 26. Vulnerabilities, threats, and defenses of the direct trading category (application layer).

threat to PADVA is a cheating notary, who quietly does not run the service periodically or runs it only sporadically. Such cheating can be revealed by clients on an ad-hoc basis, who punish notaries by the smart contract logic that causes notaries to lose their deposit.

2) Side Effects of Countermeasures: Since security aspects in notaries are inherited from the secure timestamping and identity management categories, the side effects are also inherited from these categories.

E. Direct Trading

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 goods outside of the cryptocurrency blockchain. This challenge is also referred to as the buyer/seller dilemma: "*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?*" In the direct trading category, this problem is resolved directly between the buyer and seller, without the need for a mediator, under the assumption of a trusted seller with a verified identity. For example in BIP-70 [258], the buyer first verifies the authenticity of the seller using its X.509 certificate and then issues a payment transaction.

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the direct trading category in Figure 26. The first vulnerability represents the assumption of trust in the seller. For example in BIP-70, the buyer might ask the seller to interrupt the request and get a refund but the seller may misbehave, and thus risk a reputation loss. On the other hand, this might be tolerated if the seller spoofed her identity. A mitigation technique is to use a strong means for identity verification, including assessments from reputation systems. Another attack on BIP-70 that is worth mentioning is the Silkroad trader attack [259], in which a malicious buyer might replace her refund address and then ask the seller for a refund. After a refund, the buyer might plausibly deny receipt of a refund (and ask for a refund again) due to missing authentication on the refund address. Another potential attack related to direct trading is double-spending performed as part of the selfish mining or 51% attacks – therefore, it is important to wait for enough confirmations by the seller before releasing the goods or use consensus protocols having a fast finality (see Section VI-A).

2) Side Effects and Implications of Countermeasures: Enough confirmations by the seller imply a long waiting time for the buyer before the seller releases the goods and issues a receipt for it. In particular, this might be problematic in the case of on-premise purchases. The waiting time is dependent on the time to the finality of the underlying consensus protocol, and thus a consensus protocol with a low time to finality represents a solution. However, it is the means of the consensus layer (see Section VI).

F. Escrows

Escrows address the same problem as direct trading but in contrast to direct trading, escrows do not assume a trusted seller; instead, escrows outsource the trust into the third party, referred to as a *mediator*. The mediator might actively participate in the escrow protocol or she might be involved only in the case of a dispute. According to the decentralization of the mediator, escrow protocols can be split into *single mediator* protocols and protocols with a *group-based mediator*. Goldfeder *et al.* [260] propose a few escrow protocols from both categories, which we briefly review.

1) Single Mediator: Several proposed protocols contain a single mediator and involve 2-of-3 multi-signatures for splitting the control, threshold-based signatures for improving privacy, and protocols leveraging homomorphic properties of elliptic cryptography to achieve privacy (i.e., by blinding the mediator's next address) and non-interactiveness. Another protocol combines multi-signatures with bonds deposited by a mediator to avoid DoS by the mediator. Note that blinding of the mediator's next address hides the execution of the protocol to the mediator only in the case when no dispute has arisen.

2) Group-Based Mediator: In these protocols, disputes are resolved by a majority vote. DoS attack is thwarted as long as the majority of mediators is willing to finish the execution of the protocol. The privacy of some protocols is preserved by blinding, similarly as in the case of single mediator protocols.

An example of a distributed marketplace was proposed in [261], where a marketplace contract lists the products and an escrow agent contract serves for resolution of disputes by a mediator (viz. single or group-based mediator). The authors discuss the integration of logistic parties with verified identities and reputation systems to assess these parties and mediators. OpenBazaar[48] is a distributed marketplace that uses smart contract-based escrows with 2-of-3 multi-signatures, where the mediator is agreed by the buyer and seller. A similar example is Escaroo[49] but in contrast to Openbazaar, it has its own trusted mediator. Natmin[50] is an escrow example that utilizes a public group of mediators for dispute resolution, while the

[48]https://openbazaar.org/
[49]https://escaroo.com/
[50]https://www.natmin.io/

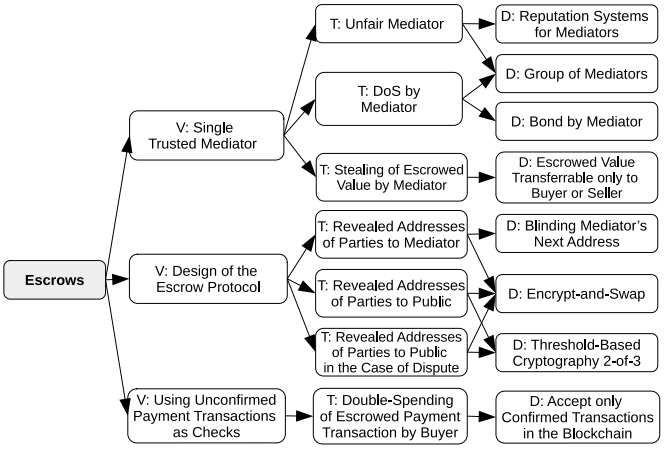


Fig. 27. Vulnerabilities, threats, and defenses of the escrows category (application layer).

reputation of the mediators is adjusted according to their votes and a result of the dispute.

3) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the escrow category in Figure 27. The first group of threats refers to a trusted mediator who represents a single-point-of-failure. The mediator might disrupt the execution of the escrow protocol or decide unfairly in the case of a dispute. The existence of these threats depends on the design of an escrow protocol. For example, in the Silk Road marketplace [262], 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.[51] However, the mediator might refuse to do so and keep the value for herself. The mitigation techniques are group-based mediators, requiring a consensus of the majority to decide on a case or requiring the mediator to put a bond into the escrow protocol. Another mitigation technique is to use reputation systems for the assessment of a single mediator.

To avoid stealing of the escrowed value by the mediator, the protocol should by design allow releasing value only to the buyer or seller (using rules of a smart contract), while assuming a timeout. For example, an early version of OpenBazaar utilized smart contracts for trading but without any timeout. As a result, many buyers did not release the funds to the seller upon successful delivery. However, when a timeout is adopted, upon its expiration, a seller can unilaterally release and acquire the funds from the escrow.

The next class of threats is related to revealing the private information about running the protocol to the public or mediator (e.g., involved parties, occurrences of disputes). The countermeasures are blinding the mediator's address, threshold-based cryptography, including its special variant *encrypt-and-swap* [260], which uses a 3-of-3 threshold signature protocol that assigns one private share to the buyer and one to the seller, while the third share is known to both parties. Both parties reveal their private share to the mediator, who, upon dispute, provides the winning party with the missing share.

Another possible threat is the double-spending of an unconfirmed escrowed payment transaction by the buyer. For example, if a (naive) escrow protocol requires a mediator to escrow a signed transaction by the buyer and release it to the blockchain only upon delivery confirmation, the buyer might not confirm delivery and perform a double-spending attack (see Section VI). As a prevention technique, unconfirmed transactions should not be accepted by the sellers at all. Moreover, we highlight that some escrow protocols (similarly as atomic swaps) are sensitive to double-spending performed by the selfish mining or 51% attacks – therefore, in these protocols, it is important to wait for enough confirmations or use consensus protocols having a fast finality (see Section VI-A).

4) Side Effects and Implications of Countermeasures: Although group-based mediators are more robust to attacks misusing a trust in a single mediator, they are more expensive to run, requiring the interaction of enough mediators with the blockchain. This in turn slows down the throughput of the escrow protocol. The extra operational overheads are also imposed by the reputation systems for single mediators. Although encoding the escrow logic into a smart contract is supported in some implementations (e.g., OpenBazaar, Escaroo), they require the deployment of a new smart contract per each trade, which is a costly option. Such logic can be implemented even within a single smart contract. Similar to direct trading, not accepting unconfirmed transactions implies a long waiting time for the buyer; this is not the case for the consensus protocols with a fast finality. Finally, using reputation systems brings their security aspects.

G. Auctions

In auctions, sellers promote the sale of their goods through blockchain while buyers place bids for them. Galal and Youssef [263] specify several desired properties of auctions:

Privacy of bids ensures that values of particular bids are not revealed to anybody before committing to them.

Posterior privacy ensures that all bids remain private after the auction ends.

Publicly verifiable correctness enables anybody to verify the results of the auction through the blockchain.

Resistance against DoS ensures that no bidder or auctioneer can prematurely abort a protocol without being penalized.

The authors of [263] instantiate the auction as a smart contract, to which, bidders submit homomorphic commitments of their sealed bids and then reveal their commitments to the auctioneer via a PK encryption. Afterward, the auctioneer deciphers the bids, determines the winner, and announces it to the public while providing zero-knowledge proofs of the correctness. As part of their other contribution [264], the same authors improved on high costs intrinsic to their former work [263] by using zk-SNARKs and its off-chain computation, which requires only a single on-chain proof verification of the whole auction process. However, in both approaches [263], [264], the auctioneer might compromise the privacy of all bids, which led the same authors to propose Trustee [265],

[51] Instead of a single mediator, the Silk Road marketplace utilized several intermediaries to increase the anonymity of the buyer and seller.

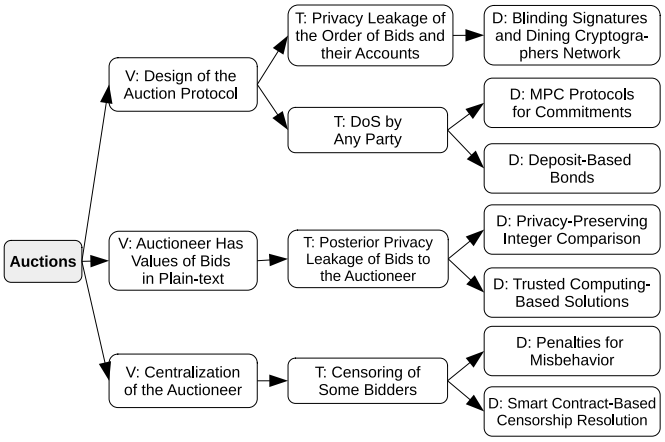


Fig. 28. Vulnerabilities, threats, and defenses of the auctions category (application layer).

an approach based on TEE. In Trustee, the bidders submit encrypted bids to the auctioneer's TEE, which confidentially evaluates a winner and generates a blockchain transaction proving it.

Strain [266] is an auction protocol that guarantees the privacy of bids against malicious bidders and, in contrast to [263], [264] also against the auctioneer. Strain is executed in four rounds (i.e., four blocks), and it assumes a semi-honest (i.e., passive) auctioneer who acts as a judge verifying the correctness of zero-knowledge proofs. Neither the 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 commit publicly to her bid, while the proposed scheme enables a majority of honest bidders to open other bidders' commitments in the case that 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 the Dining Cryptographers (DC) network.

Another approach that preserves the privacy of the bid values (but not the privacy of their order) is proposed in [267]. The protocol requires off-chain interaction for two-party computation protocol that performs a pairwise comparison of blinded bids among bidders and the auctioneer.

1) Security Threats and Mitigations: We present a taxonomy of vulnerabilities, threats, and defenses related to the auctions category in Figure 28. There are several possible issues related to privacy leakage in the auction protocols. The first privacy issue stands for revealing addresses of bidders and/or order of their bids to the public. For example, the authors of [263] do not provide anonymity of bidders, since bidders use their existing Ethereum addresses to interact with the protocol. A mitigation technique using blinding RSA signatures and the DC network was proposed in [266]; however, network-level attacks on revealing locations/IP addresses of parties remain possible (see Section V).

Since the auctioneer of some protocols (e.g., [263], [264]) sees the bidders and their bids in plain-text, she might either intentionally or accidentally (e.g., an external compromise) leak these data (and their corresponding proofs) that are attributable to particular bidders. The protection technique is to avoid the auctioneer from seeing the plain-text of the bids, and instead use privacy-preserving integer comparison (e.g., [266]) or trusted computing-based solutions (e.g., [265]).

Another threat originates from a centralized auctioneer who might censor some bidders (e.g., due to collusion with another bidder) by claiming that their bids are invalid, i.e., claiming that a commitment does not open to the sealed bid. To cope with this threat, the auction can utilize a smart contract-based resolution mechanism in which the bidder might prove the opposite by revealing her value of the bid, causing a penalization of the auctioneer [263]. Deposit-based bonds and penalization of the involved parties can be used as a protection against abortion of the protocol (i.e., DoS) by any party. For example, the protocol of [263] splits penalties to honest participants in the case of abortion by some party. Another option to cope with the abortion of the auction protocol is to use multiparty computation (MPC) for the commitment of sealed bids [266], which enables the opening of the commitment of the aborting party and thus to continue in the auction protocol.

2) Side Effects and Implications of Countermeasures: Although censoring of bids can be prevented by a smart contract-based resolution mechanism [263], it has privacy consequences since it leaks the value of the bid and its corresponding bidder. Deposit-based bonds can disincentivize the abortion of the auction protocol but they require a restart of the round. In contrast to it, MPC protocols for commitments can recover the current auction round but with additional overheads and costs imposed on a smart contract platform. In the case of using a TEE-based solution, the malicious auctioneer might perform censorship of some bidders due to collusion with other bidders. To avoid this threat, the authors of [265] propose a smart contract-based mechanism that verifies whether the set of sealed bids submitted to the smart contract corresponds to the list of bids in the proof generated by TEE. Authors also discuss another option to cope with this attack by embedding an SPV client within the TEE that would evaluate the state of the blockchain; however, this solution would impose a high memory consumption of already constrained TEE and would expose TEE to vulnerabilities presented in SPV client. Another implication of using TEE-based auction is the possibility of a local replay attack discussed in [265], where the auctioneer might provide different instances of TEE with a different subset of the bids, and hence obtain the values of particular bids. As described in [265], such a privacy threat can be prevented by a TEE specific construct called *hardware monotonic counter*, which cannot be reverted once incremented. Finally, one has to consider that a vulnerability in trusted hardware may result in the unfairness of the auction process and compromise of its privacy.

H. General Applications of Blockchains

There are many use cases of applying blockchain to a particular domain that contains mutually untrusted participants:

these participants are represented by consensus nodes executing a consensus protocol. Applications from this category focus on leveraging inherent blockchain features and sometimes even on non-inherent features (see Appendix A).

An example that uses permissioned blockchain for the management of healthcare data is proposed in [268], in which 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 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 [269]. The authors suggest that smart meters act as consensus nodes and store the data of the blockchain in their memory.[52] DistBlockNet [270] is the approach for ensuring 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. A federated permissioned blockchain used as a cloud-based data storage requiring the consensus of all nodes[53] is proposed in work [271]. 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 proofs of the first tier blockchain. Two-tier blockchain was also applied in the domain of IoT [272]. The authors of [272] 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 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 general category vary case by case and usually concern the privacy of data shared among involved parties [268], [269], [272]. Another common issue is an application of the blockchain with unrealistic assumptions about the target environment, e.g., low processing or storage performance of smart meters/IoT devices, no HW support for asymmetric cryptography, no tamper-proof means in devices producing transactions, etc. Some applications try to optimize throughput or finality of the blockchain by introducing their own consensus mechanisms (e.g., [271], [272]); 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 presented at the RSM layer (see Section VII).

X. Lessons Learned

In this Section, we summarize lessons learned concerning the security reference architecture (SRA) and its practical utilization. First, we describe the hierarchy of security dependencies among particular layers of the SRA. Second, assuming such a hierarchy, we describe a security-oriented methodology for designers of blockchain platforms and applications. Next, we summarize the design goals of particular blockchain types and discuss the security-specific features of the blockchains. Finally, we analyze observations from the incidents that occurred in practice, limitations in the literature, and we propose future research directions.

A. Hierarchy of Dependencies in the SRA

In the proposed model of the SRA, we observe that consequences of vulnerabilities presented at lower layers of the SRA are manifested in the same layers and/or at higher layers, especially at the application layer. Therefore, we refer to *security dependencies* of these layers on lower layers or the same layers, i.e., *reflexive* and *bottom-up* dependencies. We describe these two types of dependencies in the following.

1) Reflexive Dependencies: If a layer of the SRA contains some assets, it also contains a reflexive security dependency on the countermeasures presented in the same layer. It means that a countermeasure at a particular layer protects the assets presented in the same layer. For example, in the case of the consensus layer whose protocols reward consensus nodes for participation, the countermeasures against selfish mining attacks protect rewards (i.e., crypto-tokens) of consensus nodes. In the case of the RSM layer, the privacy of user identities and data is protected by various countermeasures of this layer (e.g., blinding signatures, secure multiparty computations). Another group of reflexive security dependencies is presented at the application layer. Although the application layer contains some bottom-up security dependencies (see Figure 15), we argue that with regard to the overall stacked model of the SRA they can be viewed as reflexive security dependencies of the application layer.

2) Bottom-Up Dependencies: If a layer of the SRA contains some assets, besides reflexive security dependencies, it also contains bottom-up security dependencies on the countermeasures of all lower layers. Hence, the consequences of vulnerabilities presented at lower layers of SRA might be manifested at the same layers (i.e., reflexive dependencies) but more importantly, they are manifested at higher layers, especially at the application layer. For example, context-sensitive transactions and partial solutions as countermeasures of the consensus layer can protect against front-running attacks of intra-chain DEXes, which occur at the application layer. Another example represents programming bugs in the RSM layer, which influence the correct functionality at the application layer. The eclipse attack is an example that impacts the consensus layer from the network layer – a victim consensus node operates over the attacker-controlled chain, and thus causes a loss of crypto-tokens by a consensus node and at the same time it decreases honest consensus power of the network. In turn, this might simplify selfish-mining attacks at the consensus layer, which in turn might impact the correct functionality of a blockchain-based application at the application layer. Bottom-up security dependencies are also presented in the context of the application layer, as we have already mentioned in Section VIII.

[52]These are unrealistic assumptions.

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

TABLE I
PROS AND CONS OF VARIOUS CATEGORIES WITHIN THE FIRST THREE LAYERS OF THE STACKED MODEL

Layer		Category in a Layer	Pros	Cons
Network Layer		Private Networks	• low latency, high throughput • centralized administration, ease of access control • the privacy of data and identities • meeting regulatory obligations • resilience to external attacks	• VPN is required for geographically spread participants • suitable only for permissioned blockchains • insider threat and external attacks 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 (with a small number of nodes) • 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 • high scalability • probabilistic security guarantees • lower communication overheads than BFT	• some PoS specific attacks • supports only semi-permissionless setting
		PoR+BFT	• high scalability • 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 pseoudonymous 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 anonymized to the extend of the group	• additional computation overheads for running the schemes
		MPC Blinding Signatures, Layered-Encryption	• unlinkability for all involved parties	• additional computation overheads for running the schemes
	Transactions, Protecting Data	NIZKs, Blinding Signatures, Homomorphic Encryption	• privacy of data in cryptocurrency platforms	• additional computation overheads for running the schemes
		Trusted Transaction Managers, Trusted Hardware, MPC	• privacy of data in transactions of smart contract platforms	• additional computation overheads 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

B. Methodology for Designers

A hierarchy of security dependencies in the SRA can be utilized during the design of new blockchain-based solutions. When designing a new **blockchain platform** or a new **blockchain application**, we recommend designers to specify requirements on the blockchain features (see Appendix A) and afterward analyze design options and their attack surfaces at the first three layers of the stacked model of SRA. We briefly summarize the pros and cons of particular categories within the first three layers of SRA in Table I, while security threats and mitigations are covered in Section V, Section VI, and Section VII.

TABLE II
PROS AND CONS OF SOME CATEGORIES FROM THE APPLICATION LAYER

Application Category	Subcategory	Pros	Cons
Wallets	**Server-Side Hosted Wallets**	• simplicity of control for end-users • no storage requirements for end-users	• keys stored at the server, susceptibility to the theft of keys by external or internal attacks • single-point-of-failure, availability attacks
	Client-Side Hosted Wallets	• simplicity of control for end-users • no storage requirements for end-users • keys stored locally	• single-point-of-failure, availability attacks • possibility of key theft by malware • possibility of tampering attacks
	Self-Sovereign Wallets	• keys stored locally or in a dedicated hardware device	• moderate storage requirements for end-users • more difficult control for end-users • extra device to carry in the case of hardware wallet
Exchanges	**Centralized Exchange**	• a high throughput and speed of operations • the simplicity of control for end-users • low costs for exchange transactions • trading of obscure crypto-tokens	• risk of insider threat due to centralization • external threats to exchange infrastructure • overheads for secure storage of secrets • a fee specified by the operator
	Direct Cross-Chain Exchange	• fairness of the exchange • no fee to any operator	• costs for 4 transactions of the atomic swap • user has to find the counter-order on her own • counter-orders might not exist • a lower throughput than in a centralized exchange • a higher complexity for end-users
	Cross-Chain DEX	• fairness of the exchange • order matching made by DEX • trading of obscure crypto-tokens	• costs for 4 or 6 transactions of the atomic swap • a lower throughput than in a centralized exchange • a fee specified by the operator
	Intra-Chain DEX	• fairness of the exchange • uniform finality for every pair • a high speed of operations	• a limited number of pairs that are specific to the target platform • a fee specified by the operator • costs for smart contract execution
Oracles	**Prediction Markets**	• early (close to accurate) estimation of the future event's result • decentralization	• possible conflict of interest • a limited set of data specific to a few events • a long time to obtain a final result, especially in the case of disputes
	Centralized Data Feeds	• wide range of data • fast provisioning time • handling of private parameters of requests • censorship evidence	• centralization (accidentally or intentionally wrong data) • availability issues
	Oracle Networks	• decentralization • wide range of data • fast provisioning time	• unsupported private parameters of requests • publicly visible data and requests
Filesystems	**Fully Replicated FSs with Ledger**	• a high availability • accountability and auditability	• a high storage overheads and operational costs • a high price
	Partially Replicated FSs with Ledger	• reasonably high availability • accountability and auditability • a lower price than in a full replication	• attack vectors specific to partial replication
	Partially Replicated FSs without Ledger	• reasonably high availability • a lower price than in a full replication	• a lack of native accountability and auditability • low durability due to a lack of incentives for storage
	Centralized Storage of Off-Chain Data	• a low price • accountability and auditability	• a low availability

On top of that, we recommend the designers of a new **blockchain application** to analyze particular options and their security implications at the application layer of SRA. We list the pros and cons of a few categories from the application layer in Table II,[54] while security threats and mitigation techniques of this layer are elaborated in Section VIII and Section IX. During this process, we recommend the designers to follow security dependencies of the target category on other underlying categories (see Figure 15) if their decentralized variants are used (which is a preferable option from the security point-of-view). For example, if one intends to design a decentralized reputation system, she is advised to study the security threats from the reputation system category and its recursive dependencies on e-voting, identity management, crypto-tokens & wallets, and (optionally) filesystems.

1) Divide and Conquer: If a designer of the blockchain application is also designing a blockchain platform, we recommend her to split the functionality of the solution with the divide-and-conquer approach respecting particular layers of our stacked model. In detail, if some functionality is specific to the application layer, then it should be implemented at that

[54]Note that the table contains only categories with specified subcategorizations that represent the subject to a comparison.

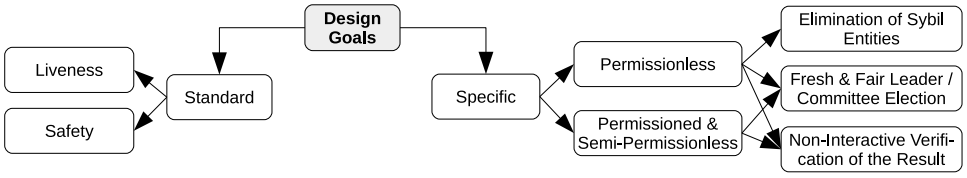


Fig. 29. Standard and specific design goals of consensus protocols.

layer. Such an approach minimizes the attack surface of a solution and enables isolating the threats to specific layers, where they are easier to protect from and reviewed by the community. A contra-example is to incorporate a part of application layer functionality/validation into the consensus layer. The consensus layer should deal only with the ordering of transactions, and it should be agnostic to the application.

Nevertheless, it is worth noting that the divide-and-conquer approach might not be suitable for some very specific cases. For example, some decentralized filesystems (see Section VIII-D) might combine data storage as an application-layer service with the proof-of-storage consensus algorithm, presented at the consensus layer. Therefore, the consensus layer also embeds a part of functionality from the application layer. However, when filesystems are in security dependencies of the target application other than filesystems, one should realize that they are usually running on a different blockchain or infrastructure than the target application, and this exception is not a concern.

C. Blockchain Types & Design Goals

We learned that the type of a blockchain (see Section III-B) implies the specific design goals of its consensus protocol (see Figure 29), which must be considered on top of the standard design goals (i.e., liveness and safety) and the inherent features (see Appendix A1) during the design of a particular blockchain platform and its consensus protocol. In the following, we elaborate on such specific design goals.

1) Permissionless Type: The first design goal is to *eliminate Sybil entities* – such elimination can be done by requiring that some amount of scarce resources is spent for extension of the blockchain, and hence no Sybil entity can participate. This implies that no pure PoS protocol can be permissionless since it never spends resources on running a consensus protocol. The next design goal is *a fresh and fair leader/committee election*, which ensures that each consensus node influences the result of a consensus commensurately to the number of scarce resources spent. Moreover, freshness avoids the prediction of the selected nodes, and therefore elected nodes cannot become the subject of targeted DoS attacks. The last design goal is the *non-interactive verification* of the consensus result by any node – i.e., any node can verify the result of the consensus based on the data presented in the blockchain.

2) 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 existing consensus node (i.e., semi-permissionless type).

3) Blockchain Types and Incentives: We observed that no application running on a *public* (permissioned) blockchain 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, notaries, secure timestamping, or reputation systems. In these blockchains, incentive schemes serve as a means for the elimination of Sybil entities, besides other purposes. 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 necessarily need crypto-tokens to operate. Misaligned incentives can cause consensus-level vulnerabilities, e.g., when it becomes profitable to drop blocks of other nodes to earn higher mining rewards [82] or transaction fees [273]. The design of incentive mechanisms is a research field by itself and we refer the reader to the work of Leonardos *et al.* [274].

D. Security-Specific Features of Blockchains

We realized that consensus protocols are the target of most financially-oriented attacks on the decentralized infrastructure of blockchains, even if such attacks might originate from the network layer (e.g., routing and eclipse attacks). The goal of these attacks is to overturn and re-order already ordered blocks while doing double-spending. Hence, the finality is the most security-critical feature of the consensus layer. The finality differs per various categories of the consensus layer. The best finality is achieved in the pure BFT protocols, and the worse finality is achieved in the single-leader-based PoR and PoS protocols. On the other hand, combinations of the BFT with PoS protocols (i.e., introducing committees) slightly deteriorate the finality of BFT in a probabilistic ratio that is commensurate to the committee size. In the case of PoR protocols with partial solutions, finality is improved as opposed to pure PoR protocols; however, it is also probabilistic, depending on the number of partial solutions.

E. Incidents in Practice

We list several incidents at each layer of the SRA in Table VI, Table V, Table IV, and Table III of Appendix. The observations about the number of different incident types vary layer by layer. In the case of the network layer, many of the attacks described in the SRA occurred or were demonstrated with a proof-of-concept. However, incidents that occurred at the consensus layer mostly contained 51% attacks with double-spending, while incidents that occurred on the application layer

were mostly caused by the exploitation of centralized components. In the case of the RSM layer, most of the incidents occurred due to bugs in smart contracts. Finally, we observed that most of the incidents that occurred at the application layer were caused due to a single-point-of-failure, e.g., centralized components or the insider threat.

Although the number of occurred incident types is low as compared to described vulnerabilities and threats, 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 observed in practice will grow.

F. Limitations in the Literature and Practice

1) Applications of Blockchains: Although the literature contains surveys and overviews [11], [12], [275] of blockchain-based applications, these works introduce only domain-oriented categorizations (i.e., categories such financial, governance, security, education, supply chain, etc.) and they do not investigate the security aspects and functionalities that these applications leverage on and whether some of the applications do not belong to the same category from the security and functionality point-of-view. To address this limitation, we provide a security-driven functionality-oriented categorization of blockchain-based applications (see Section VIII), which is agnostic to an application domain and thus can generalize different application scenarios. Furthermore, our proposed categorization enables us to model security and functionality-based dependencies among particular categories, which is not possible with state-of-the-art categorizations.

2) Centralization: Even though blockchains are meant to be fully decentralized, we have seen that this does not hold at some layers of the SRA – the network and application layers, in particular. In the network layer, some attacks are possible due to centralized DNS bootstrapping, while in the application layer a few categories utilize centralized components to ensure some functionality that cannot run on-chain or its provisioning would be too expensive and slow, which, however, forms the trade-off with the security. Some applications might depend on components from other application categories (e.g., identity management) but implementing these components in a centralized fashion, even though there exist some decentralized variants that are gaining popularity (e.g., DIDs [215] for identity management).

G. Future Research Directions

1) Fast Finality: Although finality is the most security-critical feature of the consensus layer (see Section X-D), it forms the trade-off with scalability. Therefore, we believe that the future focus of the consensus research should be in a thorough evaluation of this trade-off across various consensus protocols.

2) Network-Layer Security: We learned that a substantial body of security research in blockchains is focusing on the consensus and RSM layers since these layers are mostly identified with the blockchains. As opposed to them (see Figure 1), the network layer is not so popular even though the serious threats originating from this layer might hurt the higher layers and their assets. Therefore, a potential direction for future research lies in studying the security aspects of network protocols, their suitability for a decentralized environment, and potential improvements.

3) Privacy Preservation & Performance: All cryptographic privacy preservation techniques (see Section VII-A1) bring additional computation overhead, and thus they negatively impact the throughput of the blockchains. On the other hand, privacy-preserving solutions that are based on the trusted hardware might provide higher performance, but they rely on the manufacturer of trusted hardware and the assumption that it will not be compromised. Therefore, we believe that optimizing the trade-off between performance and privacy-preservation is an important future research direction concerning the RSM layer.

4) Security Analysis of the Application Layer: Although many references included in this study are presented in the application layer, only a very few of them analyze thoroughly security aspects of a particular application layer category or its instance. Therefore, as a future research direction, we recommend the authors of the blockchain-based applications to analyze the resistance of their applications to all known threats of a particular application category (e.g., with help of our work), while broadly think of new vulnerabilities and threats that might be specific to their application type.

5) Decentralization: Since some blockchain applications utilize centralized components while their decentralized variants already exist Section X-F, we suggest that a potential future direction for researchers and practitioners might be the concept of a fully decentralized blockchain ecosystem. Such an ecosystem might consist of only decentralized (or partially decentralized) application types, for example, the ones that we reviewed in the application layer of the SRA.

XI. Discussion

In this Section, we first discuss the versatility and modularity of the stacked model that is the proposed security reference architecture (SRA) based on. Then, we outline a few additional security aspects related to blockchains, which, for clarity and simplicity, we have not pursued throughout this article or mentioned them only tangentially. Finally, 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.

A. Stacked Model

1) Versatility: The hierarchical stacked model that is the SRA based on was already utilized in other domains before. A well-known example is the ISO/OSI model with seven layers or later derived TCP/IP model with four layers in the field of communication networks. The stacked model was also applied in cloud computing, referred to as cloud stack [276], in which, each layer represents one service model in the model's hierarchy. Nevertheless, the stacked model was also applied in the field of blockchains [16].

The versatility of the stack model allows not only for modeling the hierarchy in a particular domain but also for partitioning the corresponding security issues and their countermeasures based on the layers of the model. This was done for the ISO/OSI model [277], TCP/IP model [278], and cloud computing [8], while in this work we focus on the security threats related to blockchains and propose the SRA.

2) Modularity: The stack model of SRA enables extensions of particular categories within each layer by adding new vulnerabilities, threats, and their respective countermeasures. Likewise, the new categories can be modularly added to each of the SRA layers. Afterward, the security implications of a new category, threat, or a countermeasure within some layer should be studied with regard to particular categories in higher layers – a new category might be beneficial or detrimental to them from a security point-of-view. When introducing a new defense or mitigation technique, it is also important to evaluate its side effects and implications on the features of the blockchain that are manifested at the same and higher layers.

A general guideline for extending the SRA is to introduce only such categories that have unique features from the security point-of-view, while in the case of the application layer, functionality point-of-view should be considered as well.

B. Additional Security Aspects

1) 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[55] that are based on existing standards (e.g., [279], [280]). Examples include secure communication (i.e., network layer), the use of private keys for transaction signing (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 in the current article.

2) Semantic Bugs: We deal with semantic bugs only at the level of the RSM layer as part of the smart contracts (see Section VII-B). However, we emphasize that semantic bugs in the blockchain infrastructure may occur at each of the proposed layers, whereas in the case of the RSM layer, besides smart contracts, they may occur in compilers, interpreters, etc. In this work, we assume that the software of the blockchain-related infrastructure does not contain any programming semantic bugs at each of the layers, and it provides the expected functionality. On the other hand, we emphasize that these semantic bugs had already accounted for several incidents in the past, e.g., [281], [282], [283], [284]. To achieve safe and correct software at each of the layers, similar to the case of the RSM layer, developers and designers should utilize verification tools, testing, code reviews, audits, known design patterns, best practices, etc.

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 running on smart contract platforms inherit security aspects from the oracles category, as 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 – it inherits security aspects from the crypto-tokens and wallets category (see Section VIII-A). Another example is cross-chain communication, which is a generalization of the exchanges category, and it also inherits most of the security aspects from it.

[55]https://www.keylength.com

XII. 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.* [16]. 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.

A. Research With Flat Categorization

Bonneau *et al.* [177] 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 in its early stages of development at that time, not much is said about RSM-layer properties, and little is said about applications beyond cryptocurrencies and data storage. Similarly, Tschorsch and Scheuermann [285] and Yli-Huumo *et al.* [286] present early survey papers that focus mostly on consensus- and network-layer attacks, but they also deal with user privacy. The latter [286] 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.* [287] present a high-level overview of blockchain security threats and incidents, but the 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.* [288] 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.

B. Research With Layered or Stacked Categorization

Wang *et al.* [16] 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.* [289] 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.* [289] put double-spending attacks into application-oriented attacks, whereas 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 article deal with crypto-jacking attacks, which 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.* [290] 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 place of our RSM layer. This leads to a difference in interpretation between their framework and ours: e.g., they consider reentrancy 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 [291], [292]) 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.* [293] and consists of six layers, where the layers stand for *the application, contract, incentive, consensus, network, and a data layer*. The works of Alkhalifah *et al.* [294] and Zhu *et al.* [295] 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.* [296] focus mainly on the consensus layer but include some network-layer attacks as well (e.g., eclipse attacks and BGP hijacking).

C. Research Focusing on a Particular Layer

Finally, there is a number of survey papers that explicitly focus on specific layers: the network layer [297], the RSM layer (in particular smart contracts) [138], [298], [299], protocols of the consensus layer [101], [115], [274], [300], [301], [302], incentives at the consensus layer [303], [304], and application layer from the general standpoint [11], or with a focus on the IoT domain [305], [306], [307], [308].

XIII. Conclusion

In this article, we focused on the systematization of the knowledge about security aspects of blockchain systems, while we aimed to create a standardized model for studying vulnerabilities and security threats. We proposed a stack-modeled security reference architecture (SRA) consisting of four layers, and at each of the layers, we surveyed categories and options for their instantiation with their respective security implications and properties. We modeled particular categories as vulnerability/threat/defense graphs, which we provided as a means for reasoning about imposed security aspects. Next, we collected a sample of blockchain-related incidents that occurred in practice, which we further categorized using our proposed model. We observed that the number of incident types occurred in practice is substantially smaller than the number of described threats, especially in the consensus and application layer. In the case of the application layer, most of the incidents occurred due to exploiting a centralized component by external or internal attackers, while in the case of the consensus layer, most of the incidents occurred due to temporary violation of protocol assumptions by 51% attacks. Finally, we presented a security-oriented methodology for designers of blockchains platforms and applications, respecting the proposed SRA.

Appendix

A. 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 by 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.

Censorship Resistance: is achieved due to decentralization, and it ensures that each valid transaction is processed and included in the blockchain.

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 [22], [309], [310] or trees [23]) are built in an append-only fashion. However, the immutability of new blocks is not immediate and depends on the time to the finality of a particular consensus protocol (see Section III-C).

TABLE III
INCIDENTS THAT OCCURRED AT THE APPLICATION LAYER

Acronym / Incident	Threat / Vulnerability	Reference	Description	Impact ($)
Gatecoin (May 2016)	Centralized server compromise (single-point-of-failure)	[314]	The exchange was the victim of a man-in-the-middle attack. The malicious external party was 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)	[315]	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)	[316]	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)	[317]	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)	[318]	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	[215]	Cyber-squatters seized identities that do not belong to them nor represent themselves.	—
Front-Running on Ethereum Exchanges	Front-running in Intra-Chain Exchanges	[92]	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.	—

Availability: although distributed ledgers are highly redundant in terms of data storage (i.e., full nodes store replicated data), the main advantage of such redundancy is paid off by the 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 due 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 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 are available consensus protocols that do not waste scarce 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.

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 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 public blockchain is visible to all nodes or public, which may lead to privacy and anonymity issues. Therefore, multiple solutions increasing anonymity (e.g., ring signatures [311] in Monero) and privacy (e.g., zk-SNARKs [312] in Zcash) were proposed in the context of cryptocurrencies, while other efforts have been made in privacy-preserving smart contract platforms [131], [132].

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 be provided too.

B. Atomic Swap Protocols

1) Atomic Swap for Two Parties: 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

TABLE IV
INCIDENTS AND POSSIBLE VULNERABILITIES AT THE RSM LAYER

Acronym / Incident	Threat / Vulnerability	Reference	Description	Impact ($)	SWC Entry
DAO Attack (June 2016)	Reentrancy	[319]	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	[320]	An unchecked return value prevented the rightful compensation of the user.	300	SWC-104
RockPaperScissors	Storing secrets	[321]	A smart contract relied on a secret value that should not be visible to other users.	—	
	Integer overflow and underflow	[322]	The user's input was not properly sanitized and overflow / underflow was possible.		SWC-101
	Unexpected value received	[323]	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	[138]	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	[324]	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	[325]	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	[326]	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	[327]	The attack exploited a mismatch between the computational cost of some operations and their gas cost.	—	
Rubixi (March 2016)	Unprotected Ether withdrawal	[138]	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	[138]	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	[139]	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	[328]	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	[329]	The unprotected self destruct functionality can be exploited to destroy contracts (or libraries), and potentially freeze funds in them.	150,000,000	SWC-106

neither of them. First, this process involves an agreement on the amount and exchange rate, and second, the execution of the exchange itself.

In a centralized scenario [313], the approach is to utilize a trusted third party for the execution of the exchange. In contrast to the centralized scenario, blockchains allow us to execute such an exchange without a requirement of the trusted party. The atomic swap protocol [314] 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

TABLE V
INCIDENTS THAT OCCURRED AT THE CONSENSUS LAYER

Acronym / Incident	Threat / Vulnerability	Reference	Description	Impact ($)
Ethereum Classic (January 2019)	51% attack & double spending (violation of assumptions)	[330]	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)	[331]	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)	[332]	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)	[333]	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)	[334]	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)	[283]	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)	[335]	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	[336]	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.	—

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 Turing-incomplete smart contracts with support for hash-locks and time-locks, for clarity, we provide a description assuming Turing-complete smart contracts, requiring four transactions:

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 another party.
2) When $\mathbb{B}$ 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) Once $\mathbb{A}$ 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) Once $\mathbb{B}$ 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 expires and redeems the funds.

2) Atomic Swap for Three Parties: In the following, we outline a three-way atomic swap protocol, where party $\mathbb{A}$ wishes to sell an asset a for BTC, the party $\mathbb{B}$ wishes to buy a for ETH, and DEX $\mathbb{E}$ is inter-mediating 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) Once $\mathbb{E}$ 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) Once $\mathbb{A}$ 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 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) When $\mathbb{B}$ notices that both $HTLC_{\mathbb{E}}$ and $HTLC_{\mathbb{A}}$ have been already correctly deployed, she reveals the secret x

TABLE VI
INCIDENTS THAT OCCURRED AT THE NETWORK LAYER (PUBLIC NETWORKS)

Acronym / Incident	Threat / Vulnerability	Reference	Description	Impact ($)
Canadian Bitcoin hijack (February 2013)	BGP prefix hijacking (routing attack & eclipse attack)	[337]	Intercepted data between Bitcoin miners and Bitcoin mining pools. Tricked honest miners were mining on an attacker controlled pool.	83,000
Bitcoin deanonymization (March 2015)	Sybil attack (identity revealing attacks)	[59]	A large number of fake nodes were introduced to deanonymize client traffic.	—
MyEtherWallet hijack (April 2018)	BGP hijacking (routing attack)	[290], [291]	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	[338]	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)	[339]	An attack to partition Bitcoin network by hijacking IP prefixes.	—
Erebus (proof-of-concept)	Malicious ISP attack	[50]	ISP uses its topological advantage to launch a stealthy partition attack.	—
Eclipse attack on Bitcoin (proof-of-concept)	Un-authenticated and unreliable peers	[47]	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)	[57]	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)	[340]	Flooding a mempool with low fee transactions resulted in 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)	[341]	Mempool is flooded by high-fee transactions, preventing regular-fee transactions being processed timely.	—

as a part of the transaction sent to $HTLC_{\mathbb{A}}$. This triggers a transfer of asset a to $\mathbb{B}$.

5) Once $\mathbb{A}$ notices that x was revealed, she sends a transaction with x to $HTLC_{\mathbb{E}}$, obtaining BTC from $\mathbb{E}$.
6) Once $\mathbb{E}$ notices that x was revealed, she sends a transaction with x to $HTLC_{\mathbb{B}}$, obtaining ETH from $\mathbb{B}$.

C. Examples of Incidents

In the current section, we list several incidents at each layer of the security reference architecture. In detail, Table VI contains incidents of public networks at the network layer, Table V lists incidents of the consensus layer, Table IV focuses on the RSM layer, and Table III shows a few examples of incidents at the application layer.

ACKNOWLEDGMENT

The authors would like to thank our colleagues Pieter Hartel, Stefanos Leonardos, and Mark van Staalduinen for their valuable feedback.

REFERENCES

[1] (2019). *Enterprise Ethereum Alliance*. [Online]. Available: https://entethalliance.org/

[2] ISO. (2019). *ISO/DTR 23245: Blockchain and Distributed Ledger Technologies—Security Risks, Threats, and Vulnerabilities*. [Online]. Available: https://www.iso.org/standard/75062.html?browse=tc

[3] ISO. (2019). *ISO/CD 23257: Blockchain and Distributed Ledger Technologies—Reference Architecture Vulnerabilities*. [Online]. Available: https://www.iso.org/standard/75093.html?browse=tc

[4] L. Goasduff. (2019). *Gartner Says Blockchain Deployments Across Financial Services Ecosystems Are At Least Three Years Away*. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/09-16-2019-gartner-says-blockchain-deployments-across-financial-services-ecosystems-are-at-least-three-years-away

[5] J. Barry. (2018). *Blockchain Technology Needs Standardization*. [Online]. Available: https://medium.com/blockchain-standards/blockchain-technology-needs-standardization-596fbea2d0cf

[6] H. Zimmermann, "OSI reference model—The ISO model of architecture for open systems interconnection," *IEEE Trans. Commun.*, vol. C-28, no. 4, pp. 425–432, Apr. 1980.

[7] F. Liu *et al.*, *NIST Cloud Computing Reference Architecture*, document SP 500, NIST, Gaithersburg, MD, USA, 2011.

[8] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 843–856, 2nd Quart., 2013.

[9] Common Criteria. (2017). *Common Criteria for Information Technologysecurity Evaluation. Part 1: Introduction and General Model*. [Online]. Available: https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf

[10] I. Homoliak, S. Venugopalan, Q. Hum, and P. Szalachowski, "A security reference architecture for blockchains," in *Proc. 2nd IEEE Int. Conf. Blockchain (Blockchain)*, 2019, pp. 390–397.

[11] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telemat. Informat.*, vol. 36, pp. 55–81, Mar. 2019.

[12] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[13] J. F. Wolfswinkel, E. Furtmueller, and C. P. Wilderom, "Using grounded theory as a method for rigorously reviewing literature," *Eur. J. Inf. Syst.*, vol. 22, no. 1, pp. 45–55, 2013.

[14] R. Tahir *et al.*, "Mining on someone else's dime: Mitigating covert mining operations in clouds and enterprises," in *Proc. Int. Symp. Res. Attacks Intrusions Defenses*, 2017, pp. 287–310.

[15] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proc. 40th Annu. Symp. Found. Comput. Sci.*, 1999, pp. 120–130.

[16] W. Wang *et al.*, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019.

[17] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[18] P. Szalachowski, D. Reijsbergen, I. Homoliak, and S. Sun, "StrongChain: Transparent and collaborative proof-of-work consensus," in *Proc. 28th USENIX Security Symp. (USENIX Security)*, Santa Clara, CA, USA, Aug. 2019, pp. 819–836.

[19] Hyperledger Team. (2017). *Hyperledger Architecture, Volume 1: Consensus*. [Online]. Available: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf

[20] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (POET)," in *Proc. Int. Symp. Stabilization Safety Security Distrib. Syst.*, 2017, pp. 282–297.

[21] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proc. CRYPTO*, 2017, pp. 357–388.

[22] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 1159, 2016.

[23] Y. Sompolinsky and A. Zohar, "Accelerating bitcoin's transaction processing. Fast money grows on trees, not chains," in *Proc. IACR Cryptol. ePrint Archive*, vol. 2013, 2013, p. 881.

[24] W. Tang. (2017). *ECIP-1029: Include Uncles in Total Difficulty Calculation*. [Online]. Available: https://github.com/ethereumproject/ECIPs/pull/71

[25] A. Zamyatin, N. Stifter, P. Schindler, E. Weippl, and W. J. Knottenbelt. (2018). *FLUX: Revisiting Near Blocks for Proof-of-Work Blockchains*. [Online]. Available: https://eprint.iacr.org/2018/415/20180529:172206

[26] M. Castro *et al.*, "Practical Byzantine fault tolerance," in *Proc. OSDI*, 1999, p. 99.

[27] P.-L. Aublin, S. B. Mokhtar, and V. Quéma, "RBFT: Redundant Byzantine fault tolerance," in *Proc. IEEE ICDCS*, 2013, pp. 297–306.

[28] E. Buchman, J. Kwon, and Z. Milosevic, "The latest Gossip on BFT consensus," 2018. [Online]. Available: arxiv.abs/1807.04938.

[29] A. Kiayias and A. Russell, "Ouroboros-BFT: A simple Byzantine fault tolerant consensus protocol," in *Proc. IACR Cryptol. ePrint Arch.*, 2018, p. 1049.

[30] S. Duan, H. Meling, S. Peisert, and H. Zhang, "BChain: Byzantine replication with high throughput and embedded reconfiguration," in *Proc. OPODIS*, 2014, pp. 91–106.

[31] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proc. ACM SOSP*, 2017, pp. 51–68.

[32] P. Daian, R. Pass, and E. Shi, "Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake," in *Proc. Int. Conf. Financial Cryptogr. Data Security*. 2019, pp. 23–41.

[33] D. W. T. Hanke and M. Movahedi, "DFINITY technology overview series, consensus system," 2018. [Online]. Available: arXiv:1805.04548.

[34] *The ZILLIQA Technical Whitepaper*, ZILLIQA Team, Singapore, 2017.

[35] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE S&P*, San Francisco, CA, USA, May 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1109/SP.2018.000-5

[36] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM CCS*, 2018, pp. 931–948. [Online]. Available: https://doi.org/10.1145/3243734.3243853

[37] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Comput. Surveys*, vol. 22, no. 4, pp. 299–319, 1990.

[38] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, 1998.

[39] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX ATC*, 2014, pp. 305–319.

[40] B. M. Oki and B. H. Liskov, "Viewstamped replication: A new primary copy method to support highly-available distributed systems," in *Proc. ACM Symp. Principles Distrib. Comput.*, 1988, pp. 22–29.

[41] C. Cachin and J. A. Poritz, "Secure intrusion-tolerant replication on the internet," in *Proc. IEEE Int. Conf. Depend. Syst. Netw.*, 2002, pp. 167–176.

[42] M. L. Collins *et al.*, "Common sense guide to prevention and detection of insider threats 5th edition," in *Proc. CERT*, 2016, pp. 1–9.

[43] D. S. H. Rosenthal, P. Maniatis, M. Roussopoulos, T. J. Giuli, and M. Baker. (2004). *Notes on the Design of an Internet Adversary*. [Online]. Available: http://arxiv.org/abs/cs.DL/0411078.

[44] S. Bradshaw and L. DeNardis. (2016). *The Politicization of the Internet's Domain Name System: Implications for Internet Security, Universality, and Freedom*. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/1461444816662932

[45] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *Proc. IEEE Symp. Security Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 375–392.

[46] M. Apostolaki, G. Marti, J. Müller, and L. Vanbever, "SABRE: Protecting bitcoin against routing attacks," in *Proc. NDSS*, 2019, pp. 1–8.

[47] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *Proc. USENIX Security*, 2015, pp. 129–144.

[48] K. Wüst and A. Gervais, "Ethereum eclipse attacks," ETH Zürich, Zurich, Switerland, Rep. ethz-a-010724205, 2016.

[49] Y. Marcus, E. Heilman, and S. Goldberg, "Low-resource eclipse attacks on Ethereum's peer-to-peer network," in *Proc. IACR Cryptol. ePrint Arch.*, 2018, p. 236.

[50] M. Tran, I. Choi, G. J. Moon, A. V. Vu, and M. S. Kang, "A stealthier partitioning attack against bitcoin peer-to-peer network," in *Proc. IEEE Symp. Security Privacy (S&P)*, 2020, pp. 894–909.

[51] B. Alangot, D. Reijsbergen, S. Venugopalan, and P. Szalachowski, "Decentralized lightweight detection of eclipse attacks on bitcoin clients," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Greece, Rhodes Island, Nov. 2020, p. 1.

[52] S. Higgins. (2015). *Bitcoin Mining Pools Targeted in Wave of DDoS Attacks*. [Online]. Available: https://www.coindesk.com/bitcoin-mining-pools-ddos-attacks

[53] D. Shares. (2017). *Major DDoS Attacks Hit Bitcoin.Com*. [Online]. Available: https://news.bitcoin.com/ddos-attacks-bitcoin-com-uncensored-information/

[54] E. Arazi. (2018). *Choosing the Right DDoS Solution (Part 4): Hybrid Protection*. [Online]. Available: https://blog.radware.com/security/2018/04/choosing-the-right-ddos-solution-hybrid-protection/

[55] S. D. Lerner. (2019). *New DoS Vuln by Forcing Continuous Hard Disk Seek/Read Activity (Fixed in 0.8.0)*. [Online]. Available: https://bitcointalk.org/index.php?topic=144122.0

[56] BitcoinWiki. *Weaknesses*. Accessed: Jul. 24, 2018. [Online]. Available: https://en.bitcoin.it/wiki/Weaknesses#Denial_of_Service_.28DoS.29_attacks

[57] A. Biryukov and I. Pustogarov, "Bitcoin over tor isn't a good idea," in *Proc. IEEE Symp. Security Privacy (SP)*, San Jose, CA, USA, May 2015, pp. 122–134. [Online]. Available: https://doi.org/10.1109/SP.2015.15

[58] A. Miller *et al.*, *Discovering Bitcoin's Public Topology and Influential Nodes*, Univ. Maryland, College Park, MD, USA, 2015.

[59] G. Caffyn. (2015). *Chainalysis CEO Denies Sybil Attack on Bitcoin Network*. [Online]. Available: https://www.coindesk.com/chainalysis-ceo-denies-launching-sybil-attack-on-bitcoin-network

[60] A. Miller, A. Kosba, J. Katz, and E. Shi, "Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions," in *Proc. ACM CCS*, 2015, pp. 680–691.

[61] V. Buterin and V. Griffith, "Casper the friendly finality gadget," 2017. [Online]. Available: arxiv.abs/1710.09437.

[62] X. Yang, Y. Chen, and X. Chen, "Effective scheme against 51% attack on proof-of-work blockchain with history weighted information," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, 2019, pp. 261–265.

[63] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of BFT protocols," in *Proc. ACM CCS*, 2016, pp. 31–42.

[64] A. Boverman. (2011). *Timejacking & Bitcoin*. [Online]. Available: http://culubas.blogspot.com/2011/05/timejacking-bitcoin_802.html

[65] P. Szalachowski, "(Short paper) towards more reliable bitcoin timestamps," in *Proc. IEEE CVCBT*, 2018, pp. 1–5.

[66] S. Dexter. (2018). *1% Shard Attack Explained—Ethereum Sharding (Contd.)*. [Online]. Available: https://www.mangoresearch.co/1-shard-attack-explained-ethereum-sharding-contd/

[67] M. Bjelic. *On the Probability of 1% Attack*. Accessed: Aug. 16, 2019. [Online]. Available: https://ethresear.ch/t/on-the-probability-of-1-attack/4009/6

[68] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 17–30.

[69] E. Syta *et al.*, "Scalable bias-resistant distributed randomness," in *Proc. IEEE Symp. Security Privacy (SP)*, 2017, pp. 444–460.

[70] A. Sonnino, S. Bano, M. Al-Bassam, and G. Danezis, "Replay attacks and defenses against cross-shard consensus in sharded distributed ledgers," 2019. [Online]. Available: arxiv.abs/1901.11218.

[71] J. Long and R. Wei, "Scalable BFT consensus mechanism through aggregated signature Gossip," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, 2019, pp. 360–367.

[72] F. Silva, A. Alonso, J. Pereira, and R. Oliveira, "A comparison of message exchange patterns in BFT protocols," in *Proc. IFIP Int. Conf. Distrib. Appl. Interoperable Syst.*, 2020, pp. 104–120.

[73] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, "Proofs of space," in *Proc. CRYPTO*, 2015, pp. 1–25.

[74] S. Park, K. Pietrzak, J. Alwen, G. Fuchsbauer, and P. Gazi. (2015). *SpaceCoin: A Cryptocurrency Based on Proofs of Space*. [Online]. Available: https://eprint.iacr.org/2015/528.pdf

[75] T. Hønsi, "Spacemint—A cryptocurrency based on proofs of space," M.S. thesis, Dept. Comput. Sci., NTNU, Trondheim, Norway, 2017.

[76] K. Karantias, A. Kiayias, and D. Zindros. (2019). *Proof-of-Burn*. [Online]. Available: https://eprint.iacr.org/2019/1096.pdf

[77] P4Titan. (2014). *SlimCoin: A Peer-to-Peer Crypto-Currency With Proof-of-Burn*. [Online]. Available: ttps://github.com/slimcoin-project/slimcoin-project.github.io/blob/master/whitepaperSLM.pdf

[78] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *Proc. IEEE S&P*, 2014, pp. 475–490.

[79] "FileCoin: A decentralized storage network," Protocol Labs, San Francisco, CA, USA, Rep., 2017. [Online]. Available: https://filecoin.io/filecoin.pdf

[80] S. King and S. Nadal. (2012). *PPcoin: Peer-to-Peer Crypto-Currency With Proof-of-Stake*. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf

[81] R. Zhang and B. Preneel, "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security," in *Proc. IEEE S&P*, 2019, pp. 175–192.

[82] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, no. 7, pp. 95–102, 2018.

[83] J. Brown-Cohen, A. Narayanan, A. Psomas, and S. M. Weinberg, "Formal barriers to longest-chain proof-of-stake protocols," in *Proc. ACM Conf. Econ. Comput.*, 2019, pp. 459–473.

[84] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in Bitcoin," in *Proc. FC*, 2016, pp. 515–532.

[85] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an Eclipse attack," in *Proc. IEEE EuroSP*, 2016, pp. 305–320.

[86] R. Pass and E. Shi, "FruitChains: A fair blockchain," in *Proc. PODC*, 2017, pp. 315–324.

[87] R. Zhang and B. Preneel, "Publish or perish: A backward-compatible defense against selfish mining in bitcoin," in *Proc. CT-RSA*, 2017, pp. 277–292.

[88] 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 *Proc. USENIX Security*, 2016, pp. 279–296.

[89] A. Miller. (2013). *Feather-Forks: Enforcing a Blacklist With Sub—50% Hash Power*. [Online]. Available: https://bitcointalk.org/index.php?topic=312668.0

[90] P. R. Rizun, "Subchains: A technique to scale Bitcoin and improve the user experience," *Ledger*, to be published.

[91] 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 *Proc. 3rd Workshop Bitcoin Blockchain Res. (BITCOIN)*, 2016, pp. 1–8.

[92] P. Daian *et al.*, "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability," in *Proc. IEEE Symp. Security Privacy (SP)*, 2020, pp. 566–583.

[93] M. Rosenfeld. (2011). *Analysis of Bitcoin Pooled Mining Reward Systems*. [Online]. Available: http://arxiv.org/abs/1112.4980.

[94] Raulo. (2011). *Optimal Pool Abuse Strategy*. [Online]. Available: http://bitcoin.atspace.com/poolcheating.pdf

[95] O. Schrijvers, J. Bonneau, D. Boneh, and T. Roughgarden, "Incentive compatibility of bitcoin mining pool reward functions," in *Proc. Financial Crypto*, 2016, pp. 477–498.

[96] S. Bag, S. Ruj, and K. Sakurai, "Bitcoin block withholding attack: Analysis and mitigation," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1967–1978, Aug. 2017.

[97] S. Bag and K. Sakurai, "Yet another note on block withholding attack on bitcoin mining pools," in *Proc. Int. Conf. Inf. Security*, 2016, pp. 167–170.

[98] P2Pool.org. (2017). *P2Pool—Decentralized Bitcoin Mining Pool*. [Online]. Available: http://p2pool.org/

[99] A. Bessani, J. Sousa, and E. E. Alchieri, "State machine replication for the masses with BFT-smart," in *Proc. IEEE/IFIP DSN*, 2014, pp. 355–362.

[100] C. Cachin, "Yet another visit to paxos," IBM Res., Zürich, Switzerland, Rep. RZ3754, 2009.

[101] C. Cachin and M. Vukolić, "Blockchain consensus protocols in the wild," in *Proc. DISC*, 2017, pp. 1–16.

[102] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "HotStuff: BFT consensus with linearity and responsiveness," in *Proc. ACM Symp. Principles Distrib. Comput.*, 2019, pp. 347–356.

[103] M. Franklin, "A survey of key evolving cryptosystems," *Int. J. Security Netw.*, vol. 1, nos. 1–2, pp. 46–53, 2006.

[104] M. Bellare and S. K. Miner, "A forward-secure digital signature scheme," in *Proc. CRYPTO*, 1999, pp. 116–129.

[105] P. Gaži, A. Kiayias, and A. Russell, "Stake-bleeding attacks on proof-of-stake blockchains," in *Proc. IEEE CVCBT*, 2018, pp. 85–92.

[106] M. Drijvers, S. Gorbunov, G. Neven, and H. Wee, "Pixel: Multi-signatures for consensus," in *Proc. 29th {USENIX} Security Symp. ({USENIX} Security)*, 2020, pp. 2093–2110.

[107] QuantumMechanic. (2011). *Proof of Stake Instead of Proof of Work*. [Online]. Available: https://bitcointalk.org/index.php?topic=27787.0

[108] I. Bentov, A. Gabizon, and A. Mizrahi, "Cryptocurrencies without proof of work," in *Proc. FC*, 2016, pp. 142–157.

[109] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake," in *Proc. ACM SIGMETRICS*, 2014, p. 452.

[110] I. Bentov, R. Pass, and E. Shi, "Snow white: Provably secure proofs of stake," in *Proc. IACR Cryptol. ePrint Arch.*, 2016, p. 919.

[111] D. Karakostas and A. Kiayias, "Securing proof-of-work ledgers via checkpointing." in *Proc. IACR Cryptol. ePrint Archive*, vol. 2020, 2020, p. 173.

[112] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, "Securing proof-of-stake blockchain protocols," in *Proc. DPM*, 2017, pp. 297–315.

[113] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Proc. EUROCRYPT*, 2018, pp. 66–98.

[114] V. Buterin, *Long-Range Attacks: The Serious Problem With Adaptive Proof of Work*, Ethereum, Zug, Switzerland, May 2014.

[115] S. Bano *et al.*, "SoK: Consensus in the age of blockchains," in *Proc. 1st ACM Conf. Adv. Financ. Technol. (AFT)*, Zurich, Switzerland, Oct. 2019, pp. 183–198.

[116] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar, "A survey on privacy protection in blockchain system," *J. Netw. Comput. Appl.*, vol. 126, pp. 45–58, Jan. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804518303485

[117] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in bitcoin P2P network," in *Proc. ACM CCS*, 2014, pp. 15–29.

[118] I. Pustogarov, "Deanonymisation techniques for Tor and bitcoin," Ph.D. dissertation, Comput. Sci., Univ. Luxembourg, Luxembourg City, Luxembourg, 2015.

[119] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn, "An empirical analysis of anonymity in zcash," in *Proc. USENIX Security*, 2018, pp. 463–477.

[120] M. Möser *et al.*, "An empirical analysis of traceability in the Monero blockchain," in *Proc. PETS*, vol. 2018, 2018, p. 3.

[121] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "MixCoin: Anonymity for bitcoin with accountable mixes," in *Proc. FC*, 2014, pp. 486–504.

[122] L. Valenta and B. Rowan, "BlindCoin: Blinded, accountable mixes for bitcoin," in *Financial Crypto*, M. Brenner, N. Christin, B. Johnson, and K. Rohloff, Eds. Berlin, Germany: Springer, 2015.

[123] G. Maxwell, "CoinJoin: Bitcoin privacy for the real world," in *Proc. Bitcoin Forum*, 2013, p. 9.

[124] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "CoinShuffle: Practical decentralized coin mixing for bitcoin," in *Proc. ESORICS*, 2014, pp. 345–364.

[125] J. H. Ziegeldorf, R. Matzutt, M. Henze, F. Grossmann, and K. Wehrle, "Secure and anonymous decentralized bitcoin mixing," *Future Gener.*

Comput. Syst., vol. 80, pp. 448–466, Mar. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X16301297

[126] S. Noether, "Ring signature confidential transactions for monero," in *Proc. IACR Cryptol. ePrint Arch.*, 2015, p. 1098.

[127] I. Miers, C. Garman, M. Green, and A. D. Rubin, "ZeroCoin: Anonymous distributed e-cash from bitcoin," in *Proc. IEEE Symp. Security Privacy*, 2013, pp. 397–411.

[128] E. B. Sasson *et al.*, "ZeroCash: Decentralized anonymous payments from bitcoin," in *Proc. IEEE S&P*, 2014, pp. 459–474.

[129] E. Heilman, F. Baldimtsi, and S. Goldberg, "Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions," in *Proc. FC*, 2016, pp. 43–60.

[130] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "BulletProofs: Short proofs for confidential transactions and more," in *Proc. IEEE Symp. Security Privacy (SP)*, 2018, pp. 315–334.

[131] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE S&P*, 2016, p. 10.

[132] R. Cheng *et al.*, "EKIDEN: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *Proc. IEEE Eur. Symp. Security Privacy (EuroS&P)*, 2019, pp. 185–200.

[133] G. Zyskind, O. Nathan, and A. Pentland, "ENIGMA: Decentralized computation platform with guaranteed privacy," 2015. [Online]. Available: arxiv/abs/1506.03471.

[134] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, "ZETHER: Towards privacy in a smart contract world," in *Proc. IACR Cryptol. ePrint Archive*, vol. 2019, 2019, p. 191. [Online]. Available: https://eprint.iacr.org/2019/191

[135] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero," in *Proc. Eur. Symp. Res. Comput. Security*, 2017, pp. 456–474.

[136] Zeppelin Solutions. (2017). *Serpent Compiler Audit*. [Online]. Available: https://docs.google.com/document/d/1_PqXuAkvgUAOG3jbBvaUvqN6W90eJ3N4IdTLNMRAijo/edit#heading=h.pe41jxc4c6xs

[137] F. Schrans, S. Eisenbach, and S. Drossopoulou, "Writing safe smart contracts in Flint," in *Proc. Conf. Companion 2nd Int. Art Sci. Eng. Program.*, Nice, France, Apr. 2018, pp. 218–219. [Online]. Available: https://doi.org/10.1145/3191697.3213790

[138] L. Valenta and B. Rowan, "Blindcoin: Blinded, accountable mixes for bitcoin," in *Proc. Financial Cryptogr. Data Security (FC) Int. Workshops BITCOIN WAHC Wearable* (Lecture Notes in Computer Science), vol. 8976. San Juan, Puerto Rico, Jan. 2015, pp. 112–126.

[139] A. Manning. (2018). *Solidity Security: Comprehensive List of Known Attack Vectors and Common Anti-Patterns*. [Online]. Available: shttps://blog.sigmaprime.io/solidity-security.html

[140] Aion Foundation. *List of Historical Vulnerabilities*. Accessed: Aug. 16, 2019. [Online]. Available: https://learn.aion.network/docs/list-of-historical-big-vulnerabilities

[141] SmartContractSecurity. (2019). *Smart Contract Weakness Classification Registry*. [Online]. Available: https://github.com/SmartContractSecurity/SWC-registry/

[142] Trailofbits. *Awesome Ethereum Security*. Accessed: Aug. 11, 2018. [Online]. Available: https://github.com/trailofbits/awesome-ethereum-security

[143] R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, and A. Singh, "Empirical vulnerability analysis of automated smart contracts security testing on blockchains," in *Proc. 28th Annu. Int. Conf. Comput. Sci. Softw. Eng.*, 2018, pp. 103–113.

[144] Consensys. (2016). *Security Tools*. [Online]. Available: https://consensys.github.io/smart-contract-best-practices/security_tools/

[145] S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov, "SmartCheck: Static analysis of ethereum smart contracts," in *Proc. IEEE/ACM 1st Int. Workshop Emerg. Trends Softw. Eng. Blockchain (WETSEB)*, 2018, pp. 9–16.

[146] R. Dua. (Feb. 11, 2019). *Solium Documentation Release 1.0.0*. [Online]. Available: https://media.readthedocs.org/pdf/solium/latest/solium.pdf

[147] J. Chang, B. Gao, H. Xiao, J. Sun, Y. Cai, and Z. Yang, "sCompile: Critical path identification and analysis for smart contracts," in *Proc. Formal Methods Softw. Eng. 21th Int. Conf. Formal Eng. Methods (ICFEM)*, Shenzhen, China, Nov. 2019, pp. 286–304.

[148] P. Hartel and M. van Staalduinen, "Truffle tests for free – replaying Ethereum smart contracts for transparency," iTRust, Singapore Univ. Technol. Design, Singapore, Rep. 1907.09208, Jul. 2019. [Online]. Available: https://arxiv.org/abs/1907.09208.

[149] M. Sutton, A. Greene, and P. Amini, *Fuzzing: Brute Force Vulnerability Discovery*. Upper Saddle River, NJ, USA: Pearson Educ., 2007.

[150] B. Jiang, Y. Liu, and W. Chan, "ContractFuzzer: Fuzzing smart contracts for vulnerability detection," in *Proc. 33rd ACM/IEEE Int. Conf. Autom. Softw. Eng.*, 2018, pp. 259–269.

[151] V. Wüstholz and M. Christakis. (2019). *Harvey: A Greybox Fuzzer for Smart contracts*. [Online]. Available: http://arxiv.org/abs/1905.06944.

[152] J. C. King, "Symbolic execution and program testing," *Commun. ACM*, vol. 19, no. 7, pp. 385–394, 1976. [Online]. Available: https://doi.org/10.1145/360248.360252

[153] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli, and M. Vechev, "Securify: Practical security analysis of smart contracts," in *Proc. ACM CCS*, 2018, pp. 67–82.

[154] Trailofbits. (2019). *Manticore Documentation Release 0.1.0*. [Online]. Available: https://media.readthedocs.org/pdf/manticore/latest/manticore.pdf

[155] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM CCS*, 2016, pp. 254–269.

[156] C. F. Torres, J. Schütte, and R. State, "OSIRIS: Hunting for integer bugs in Ethereum smart contracts," in *Proc. 34th Annu. Comput. Security Appl. Conf. (ACSAC)* San Juan, PR, USA, Dec. 2018, pp. 664–676. [Online]. Available: https://doi.org/10.1145/3274694.3274737

[157] A. Mavridou and A. Laszka, "Designing secure ethereum smart contracts: A finite state machine based approach," in *Proc. Financ. Cryptography Data Security 22nd Int. Conf. (FC)*, Nieuwpoort, Curaçao, Feb./Mar. 2018, pp. 523–540. [Online]. Available: https://doi.org/10.1007/978-3-662-58387-6_28

[158] E. Hildenbrandt *et al.*, "KEVM: A complete semantics of the Ethereum virtual machine," in *Proc. CSF*, 2018, pp. 204–217.

[159] T. Abdellatif and K. Brousmiche, "Formal verification of smart contracts based on users and blockchain behaviors models," in *Proc. 9th IFIP Int. Conf. New Technol. Mobility Security (NTMS)*, Paris, France, Feb. 2018, pp. 1–5. [Online]. Available: https://doi.org/10.1109/NTMS.2018.8328737

[160] Y. Murray and D. A. Anisi, "Survey of formal verification methods for smart contracts on blockchain," in *Proc. 10th IFIP Int. Conf. New Technol. Mobility Security (NTMS)*, Jun. 2019, pp. 1–6. [Online]. Available: https://doi.org/10.1109/NTMS.2019.8763832

[161] X. Bai, Z. Cheng, Z. Duan, and K. Hu, "Formal modeling and verification of smart contracts," in *Proc. 7th Int. Conf. Softw. Comput. Appl. (ICSCA)*, Feb. 2018, pp. 322–326. [Online]. Available: https://doi.org/10.1145/3185089.3185138

[162] Z. Nehai, P. Piriou, and F. F. Daumas, "Model-checking of smart contracts," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber Phys. Soc. Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul./Aug. 2018, pp. 980–987. [Online]. Available: https://doi.org/10.1109/Cybermatics_2018.2018.00185

[163] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, "ZEUS: Analyzing safety of smart contracts," in *Proc. 25th Annu. Netw. Distrib. Syst. Security Symp. (NDSS)*, San Diego, CA, USA, Feb. 2018, pp. 1–9. [Online]. Available: http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_09–1_Kalra_paper.pdf

[164] I. Grishchenko, M. Maffei, and C. Schneidewind, "A semantic framework for the security analysis of Ethereum smart contracts," in *Proc. Principles Security Trust 7th Int. Conf. (POST) Eur. Joint Conf. Theory Practice Softw. (ETAPS)*, Thessaloniki, Greece, Apr. 2018, pp. 243–269.

[165] Y. Zhou, D. Kumar, S. Bakshi, J. Mason, A. Miller, and M. Bailey, "ERAYs: Reverse engineering Ethereum's opaque smart contracts," in *Proc. USENIX Security*, 2018, pp. 1371–1385.

[166] M. Suiche, "Porosity: A decompiler for blockchain-based smart contracts bytecode," in *Proc. DEF CON*, vol. 25, 2017, p. 11.

[167] I. Nikolić, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor, "Finding the greedy, prodigal, and suicidal contracts at scale," in *Proc. 34th Annu. Comput. Security Appl. Conf. (ACSAC)*, San Juan, PR, USA, Dec. 2018, pp. 653–663.

[168] N. Grech *et al.*, "MADMAX: surviving out-of-gas conditions in Ethereum smart contracts," in *Proc. ACM Program. Lang.*, vol. 2, 2018, pp. 1–27. [Online]. Available: https://doi.org/10.1145/3276486

[169] L. Brent *et al.* (2018). *Vandal: A Scalable Security Analysis Framework for Smart Contracts*. [Online]. Available: http://arxiv.org/abs/1809.03981.

[170] J. Krupp and C. Rossow, "TEETHER: Gnawing at Ethereum to automatically exploit smart contracts," in *Proc. USENIX Security*, 2018, p. 9.

[171] S. Popejoy. (2016). *The Pact Smart Contract Language*. [Online]. Available: http://kadena.io/docs/Kadena-PactWhitepaper.pdf

[172] I. Sergey, V. Nagaraj, J. Johannsen, A. Kumar, A. Trunov, and K. C. G. Hao, "Safer smart contract programming with scilla," *Proc. ACM Program. Lang.*, vol. 3, 2019, pp. 1–30. [Online]. Available: https://doi.org/10.1145/3360611

[173] R. Klomp and A. Bracciali, "On symbolic verification of Bitcoin's script language," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology (ESORICS)* (Lecture Notes in Computer Science), J. Garcia-Alfaro, J. Herrera-JoancomartÃ■, G. Livraga, R. Rios, Eds. Cham, Switzerland: Springer, Sep. 2018, pp. 38–56.

[174] R. O'Connor, "Simplicity: A new language for blockchains," in *Proc. Workshop Program. Lang. Anal. Security (PLAS@CCS)*, Dallas, TX, USA, Oct. 2017, pp. 107–120. [Online]. Available: https://doi.org/10.1145/3139337.3139340

[175] MME. (2018). *Conceptual Framework for Legal and Risk Assessment of Crypto Tokens*. [Online]. Available: https://www.mme.ch/fileadmin/files/documents/180501_BCP_Framework_for_Assessment_of_Crypto_Tokens_-_Block_2.pdf

[176] S. Eskandari, J. Clark, D. Barrera, and E. Stobert, "A first look at the usability of bitcoin key management," 2018. [Online]. Available: arxiv.abs/1802.04351.

[177] 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 *Proc. IEEE S&P*, 2015, pp. 104–121.

[178] I. Homoliak, D. Breitenbacher, O. Hujnak, P. Hartel, A. Binder, and P. Szalachowski, "SmartOTPs: An air-gapped 2-factor authentication for smart-contract wallets," in *Proc. AFT*, 2020, pp. 145–162.

[179] Wolfie Zhao. (2018). *Bithumb $31 Million Crypto Exchange Hack: What We Know (And Don't)*. [Online]. Available: https://www.coindesk.com/bithumb-exchanges-31-million-hack-know-dont-know/

[180] Rachel Abrams and Nathaniel Popper. (2014). *Trading Site Failure Stirs Ire and Hope for Bitcoin*. [Online]. Available: https://dealbook.nytimes.com/2014/02/25/trading-site-failure-stirs-ire-and-hope-for-bitcoin/

[181] Reuters. (2016). *Bitcoin Worth $72M was Stolen in Bitfinex Exchange Hack in Hong Kong*. [Online]. Available: http://fortune.com/2016/08/03/bitcoin-stolen-bitfinex-hack-hong-kong/

[182] Dell SecureWorks. (2015). *Cryptocurrency-Stealing Malware Landscape Dell SecureWorks*. [Online]. Available: http://www.opensource.im/cryptocurrency/cryptocurrency-stealing-malware-landscape-dell-secureworks.php

[183] A. Peyton. (2017). *Cyren Sounds Siren over Bitcoin Siphon Scam FinTech Futureless*. [Online]. Available: https://www.bankingtech.com/2017/01/cyren-sounds-siren-over-bitcoin-siphon-scam/

[184] S. Goldfeder *et al.* (2015). *Securing Bitcoin Wallets Via a New DSA/ECDSA Threshold Signature Scheme*. 2015. [Online]. Available: http://stevengoldfeder.com/papers/threshold_sigs.pdf

[185] M. Westerkamp and J. Eberhardt, "zkRelay: Facilitating sidechains using zkSNARK-based chain-relays," *Contract*, vol. 1, no. 2, p. 3, 2020.

[186] M. Herlihy, "Atomic cross-chain swaps," in *Proc. ACM Symp. Principles Distrib. Comput.*, 2018, pp. 245–254.

[187] W. Warren and A. Bandeali. (2017). *0x: An Open Protocol for Decentralized Exchange on the Ethereum Blockchain*. [Online]. Available: https://github.com/0xProject/whitepaper

[188] H. Berg *et al.*, "Hanson's automated market maker," *J. Prediction Markets*, vol. 3, no. 1, pp. 45–59, 2009.

[189] N. Johnson. (2016). *EULER: The Simplest Exchange and Token Currency*. [Online]. Available: https://www.reddit.com/r/ethereum/comments/54l32y/euler_the_simplest_exchange_and_currency/

[190] E. Hertzog, G. Benartzi, and G. Benartzi. (2017). *Bancor Protocol: A Hierarchical Monetary System and the Foundation of a Global Decentralized Autonomous Exchange*. [Online]. Available: https://www.regulus.com/wp-content/uploads/2018/11/BancorProtocolWhitepaperPublicDraftv0.7.pdf

[191] A. Zamyatin *et al.*, "SoK: Communication across distributed ledgers," in *Proc. IACR Cryptol. ePrint Archive*, vol. 2019, 2019, p. 1128.

[192] K. Oosthoek and C. Doerr, "From hodl to heist: Analysis of cyber security threats to bitcoin exchanges," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, 2020, pp. 1–9.

[193] I. Bentov *et al.*, "TesserAct: Real-time cryptocurrency exchange using trusted hardware," in *Proc. IACR Cryptol. ePrint Archive*, vol. 2017, 2017, p. 1153.

[194] J. Poon and T. Dryja. (2016). *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. [Online]. Available: https://coinrivet.com/research/papers/the-bitcoin-lightning-network-scalable-off-chain-instant-payments/.

[195] G. Avarikioti, F. Laufenberg, J. Sliwinski, Y. Wang, and R. Wattenhofer, "Towards secure and efficient payment channels," 2018. [Online]. Available: arxiv.abs/1811.12740.

[196] P. McCorry, S. Bakshi, I. Bentov, A. Miller, and S. Meiklejohn. (2018). *PISA: Arbitration Outsourcing for State Channels*. [Online]. Available: https://eprint.iacr.org/2018/582

[197] B. Liu, P. Szalachowski, and S. Sun, "Fail-safe watchtowers and short-lived assertions for payment channels," in *Proc. 15th ACM ASIA Conf. Comput. Commun. Security (AsiaCCS)*, 2020, p. 9.

[198] P. Szalachowski, "PADVA: A blockchain-based TLS notary service," in *Proc. 25th IEEE Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Tianjin, China, Dec. 2019, pp. 836–843. [Online]. Available: https://doi.org/10.1109/ICPADS47876.2019.00124

[199] J. Guarnizo and P. Szalachowski, "PDFS: Practical data feed service for smart contracts," in *Proc. 24th Eur. Symp. Res. Comput. Security Comput. Security (ESORICS)*, Sep. 2019, pp. 767–789.

[200] S. Ellis, A. Juels, and S. Nazarov. (2017). *ChainLink: A Decentralized Oracle Network*. [Online]. Available: https://link.smartcontract.com/whitepaper

[201] J. Peterson and J. Krug, "AUGUR: A decentralized, open-source platform for prediction markets," 2015. [Online]. Available: arxiv.abs/1501.01042.

[202] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town Crier: An authenticated data feed for smart contracts," in *Proc. ACM CCS*, 2016, pp. 1–8.

[203] A. S. de Pedro, D. Levi, and L. I. Cuende, "WITNET: A decentralized oracle network protocol," 2017. [Online]. Available: arXiv:1711.09756.

[204] S. A. Crosby and D. S. Wallach, "Efficient data structures for tamper-evident logging." in *Proc. USENIX Security Symp.*, 2009, pp. 317–334.

[205] Proovable Team. (2019). *The Provable Blockchain Oracle for Modern DApps*. [Online]. Available: https://www.oraclize.it/

[206] S. Sen and J. Wang. (Apr. 2004). *Analyzing Peer-to-Peer Traffic Across Large Networks*. [Online]. Available: https://doi.org/10.1109/TNET.2004.826277

[207] H. Kopp, C. Bösch, and F. Kargl, "KopperCoin—A distributed file storage with financial incentives," in *Proc. Int. Conf. Inf. Security Pract. Exp.*, 2016, pp. 79–93.

[208] J. Benet, D. Dalrymple, and N. Greco, *Proof of Replication*, Protocol Labs, San Francisco, CA, USA, Jul. 2017.

[209] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention," in *Proc. IACR Cryptol. ePrint Archive*, vol. 2014, 2014, p. 803.

[210] Z. Wilcox-O'Hearn. (2003). *Names: Decentralized, Secure, Human-Meaningful: Choose Two*. [Online]. Available: https://web.archive.org/web/20011020191610/http://zooko.com/distnames.html

[211] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Bootstrapping trust in distributed systems with blockchains," *USENIX Mag.*, vol. 41, no. 3, pp. 1–7, 2016.

[212] C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, and M. Sena. (2016). *uPort: A Platform for Self-Sovereign Identity*. [Online]. Available: http://blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf

[213] Shocard Inc. (2017). *Shocard: Identity Management Verified Using the Blockchain*. [Online]. Available: http://shocard.com/wp-content/uploads/2018/01/ShoCard-Whitepaper-Dec13–2.pdf

[214] A. Tobin and D. Reed. (2016). *The Inevitable Rise of Self-Sovereign Identity*. [Online]. Available: https://sovrin.org/wp-content/uploads/2018/03/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf

[215] W3C Community. (2019). *Decentralized Identifiers (DIDs)*. [Online]. Available: https://w3c-ccg.github.io/did-spec/

[216] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An empirical study of Namecoin and lessons for decentralized namespace design," in *Proc. WEIS*, 2015, pp. 1–21.

[217] J. Clark and A. Essex, "CommitCoin: Carbon dating commitments with bitcoin," in *Proc. Int. Conf. Financ. Cryptography Data Security*, 2012, pp. 390–398.

[218] B. Gipp, N. Meuschke, and A. Gernandt, "Decentralized trusted timestamping using the crypto currency bitcoin," 2015. [Online]. Available: arxiv.abs/1502.04015.

[219] A. Kiayias and M. Yung, "Self-tallying elections and perfect ballot secrecy," in *Proc. 5th Int. Workshop Practice Theory Public Key Cryptosyst. (PKC)*, vol. 2274. Paris, France, Feb. 2002, pp. 141–158.

[220] J. Groth, "Efficient maximal privacy in boardroom voting and anonymous broadcast," in *Proc. 8th Int. Conf. Financial Cryptogr. (FC)*, vol. 3110. Key West, FL, USA, Feb. 2004. pp. 90–104.

[221] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981. [Online]. Available: http://doi.acm.org/10.1145/358549.358563

[222] J. D. Cohen and M. J. Fischer, "A robust and verifiable cryptographically secure election scheme," in *Proc. 26th Annu. Symp. Found. Comput. Sci. (SFCS)*, 1985, pp. 372–382. [Online]. Available: https://doi.org/10.1109/SFCS.1985.2

[223] P. Boucher, *What If Blockchain Technology Revolutionised Voting*, Eur. Parliament, Brussels, Belgium, 2016.

[224] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Proc. 21st Int. Conf. Financial Cryptography Data Security (FC)*, Sliema, Malta, Apr. 2017, pp. 357–375.

[225] W. Zhang *et al.*, "A privacy-preserving voting protocol on blockchain," in *Proc. 11th IEEE Int. Conf. Cloud Comput. (CLOUD)*, San Francisco, CA, USA, Jul. 2018, pp. 401–408.

[226] S. Venugopalan, I. Homoliak, Z. Li, and P. Szalachowski, "BBB-voting: 1-out-of-*k*blockchain-based boardroom voting," 2020. [Online]. Available: arxiv.abs/2010.09112.

[227] Y. Li, W. Susilo, G. Yang, Y. Yu, D. Liu, and M. Guizani, "A blockchain-based self-tallying voting scheme in decentralized IoT," 2019. [Online]. Available: arxiv.abs/1902.03710.

[228] J. Benaloh and D. Tuinstra, "Receipt-free secret-ballot elections (extended abstract)," in *Proc. 26th Annu. ACM Symp. Theory Comput. (STOC)*, 1994, pp. 544–553. [Online]. Available: http://doi.acm.org/10.1145/195058.195407

[229] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard, "Practical multi-candidate election system," in *Proc. 20th Annu. ACM Symp. Principles Distrib. Comput. (PODC)*, 2001, pp. 274–283.

[230] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme—A practical solution to the implementation of a voting booth," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn. Adv. Cryptol. (EUROCRYPT)*, vol. 921. Saint-Malo, France, May 1995, pp. 393–403.

[231] R. Canetti and R. Gennaro, "Incoercible multiparty computation," in *Proc. 37th Conf. Found. Comput. Sci.*, Oct. 1996, pp. 504–513.

[232] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, "Deniable encryption," in *Proc. 17th Annu. Int. Cryptol. Conf. Adv. Cryptol. (CRYPTO)*, vol. 1294. Santa Barbara, CA, USA, Aug. 1997, pp. 90–104.

[233] D. Khader, B. Smyth, P. Y. A. Ryan, and F. Hao, "A fair and robust voting system by broadcast," in *Proc. 5th Int. Conf. Electron. Voting (EVOTE)*, Jul. 2012, pp. 285–299. [Online]. Available: https://dl.gi.de/20.500.12116/18220.

[234] A. Schaub, R. Bazin, O. Hasan, and L. Brunie, "A trustless privacy-preserving reputation system," in *Proc. IFIP Int. Conf. ICT Syst. Security Privacy Protect.*, 2016, pp. 398–411.

[235] D. Carboni. (2015). *Feedback Based Reputation on Top of the Bitcoin Blockchain*. [Online]. Available: http://arxiv.org/abs/1502.01504

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

[237] A. Grech and A. F. Camilleri, "Blockchain in education," Publ. Office Eur. Union, Rep., 2017.

[238] P. Buneman, S. Khanna, and T. Wang-Chiew, "Why and where: A characterization of data provenance," in *Proc. Int. Conf. Database Theory*, 2001, pp. 316–330.

[239] N. Kshetri, "1 blockchain's roles in meeting key supply chain management objectives," *Int. J. Inf. Manag.*, vol. 39, pp. 80–89, Apr. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0268401217305248

[240] Deloitte. (2017). *Continuous Interconnected Supply Chain Using Blockchain and Internet-of-Things in Supply Chain Traceability*. [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/lu/Documents/technology/lu-blockchain-internet-things-supply-chain-traceability.pdf

[241] H. M. Kim and M. Laskowski, "Toward an ontology-driven blockchain design for supply-chain provenance," *Intell. Syst. Account. Finance Manag.*, vol. 25, no. 1, pp. 18–27, 2018.

[242] S. Shetty, V. Red, C. Kamhoua, K. Kwiat, and L. Njilla, "Data provenance assurance in the cloud using blockchain," in *Proc. SPIE Defense Security Conf.*, 2017, p. 102060I.

[243] C. Jaag and C. Bach, "Blockchain technology and cryptocurrencies: Opportunities for postal financial services," in *The Changing Postal and Delivery Sector*, M. Crew, P. L. Parcu, and T. Brennan, Eds. Heidelberg, Germany: Springer, 2017, pp. 205–221.

[244] T. Hardjono and N. Smith, "Cloud-based commissioning of constrained devices using permissioned blockchains," in *Proc. 2nd ACM Int. Workshop IoT Privacy Trust Security*, 2016, pp. 29–36.

[245] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for IoT data," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 468–475.

[246] D. Bhowmik and T. Feng, "The multimedia blockchain: A distributed and tamper-proof media transaction framework," in *Proc. 22nd Int. Conf. Digit. Signal Process. (DSP)*, Aug. 2017, pp. 1–5.

[247] A. Tomescu and S. Devadas, "CATENA: Efficient non-equivocation via bitcoin," in *Proc. IEEE Symp. Security Privacy (SP)*, May 2017, pp. 393–409.

[248] M. Al-Bassam and S. Meiklejohn, "CONTOUR: A practical system for binary transparency," in *Proc. Data Privacy Manag. Cryptocurrencies Blockchain Technol.*, 2018, pp. 94–110.

[249] 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 *Proc. ACM Asia Conf. Comput. Commun. Security (AsiaCCS)*, Auckland, New Zealand, Jul. 2019, pp. 479–484.

[250] I. Homoliak and P. Szalachowski, "AQUAREUM: A centralized ledger enhanced with blockchain and trusted computing," 2020. [Online]. Available: arxiv.abs/2005.13339.

[251] R. Paccagnella *et al.*, "CUSTOS: Practical tamper-evident auditing of operating systems using trusted execution," in *Proc. Symp. Netw. Distrib. Syst. Security (NDSS)*, 2020, pp. 1–9.

[252] National Notary Association, CA. (2019). *What Is Notarization?* [Online]. Available: https://www.nationalnotary.org/knowledge-center/about-notaries/what-is-notarization

[253] Heleness. (2018). *Blockchain-Based Notarization Platform on Ethereum*. [Online]. Available: https://ethresear.ch/t/blockchain-based-notarization-platform-on-ethereum/5326

[254] ERC-725 Alliance. (2018). *ERC-725: Ethereum Identity Standard*. [Online]. Available: https://erc725alliance.org/

[255] K. Rantos, G. Drosatos, K. Demertzis, C. Ilioudis, and A. Papanikolaou, "Blockchain-based consents management for personal data processing in the IoT ecosystem." in *Proc. ICETE*, 2018, pp. 738–743.

[256] A. Mizrahi. (2015). *A Blockchain-Based Property Ownership Recording System*. [Online]. Available: https://static1.squarespace.com/static/5e26f18cd5824c7138a9118b/t/5e3c1c7d6d5ff061da34da80/1580997757765/A-blockchain-based-property-registry.pdf

[257] B. Notheisen, J. B. Cholewa, and A. P. Shanmugam, "Trading real-world assets on blockchain," *Bus. Inf. Syst. Eng.*, vol. 59, no. 6, pp. 425–440, 2017.

[258] G. Andresen and M. Hearn. (2016). *BIP-70*. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0070.mediawiki

[259] P. McCorry, S. F. Shahandashti, and F. Hao, "Refund attacks on bitcoin's payment protocol," in *Proc. Int. Conf. Financ. Cryptography Data Security*, 2016, pp. 581–599.

[260] S. Goldfeder, J. Bonneau, R. Gennaro, and A. Narayanan, "Escrow protocols for cryptocurrencies: How to buy physical goods using bitcoin," in *Proc. Int. Conf. Financ. Cryptography Data Security*, 2017, pp. 321–339.

[261] O. R. Kabi and V. N. Franqueira, "Blockchain-based distributed marketplace," in *Proc. Int. Conf. Bus. Inf. Syst.*, 2018, pp. 197–210.

[262] N. Christin, "Traveling the silk road: A measurement analysis of a large anonymous online marketplace," in *Proc. ACM 22nd Int. Conf. World Wide Web*, 2013, pp. 213–224.

[263] H. S. Galal and A. M. Youssef, "Verifiable sealed-bid auction on the Ethereum blockchain," in *Proc. Int. Conf. Financ. Cryptography Data Security*, 2018, pp. 265–278.

[264] H. Galal and A. Youssef, "Succinctly verifiable sealed-bid auction smart contract," in *Proc. Data Privacy Manag. Cryptocurrencies Blockchain Technol.*, 2018, pp. 3–19.

[265] H. S. Galal and A. M. Youssef, "Trustee: Full privacy preserving vickrey auction on top of ethereum," in *Proc. Financial Cryptography Data Security (FC) Int. Workshops*, Feb. 2019, pp. 190–207.

[266] E.-O. Blass and F. Kerschbaum, "STRAIN: A secure auction for blockchains," in *Proc. Eur. Symp. Res. Comput. Security*, 2018, pp. 87–110.

[267] J. Ma, B. Qi, and K. Lv, "Fully private auctions for the highest bid," in *Proc. ACM Turing Celebration Conf. China*, 2019, p. 64.

[268] 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 Comput.*, vol. 5, no. 1, pp. 31–37, Mar. 2018.

[269] 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 Trans. Smart Grid*, vol. 10, no. 3, pp. 3162–3173, Mar. 2018.

[270] P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, "DistBlockNet: A distributed blockchains-based secure SDN architecture for IoT networks," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 78–85, Sep. 2017.

[271] 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 *Proc. 1st Ital. Conf. Cybersecurity (ITASEC)*, Venice, Italy, Jan. 2017, pp. 146–155.

[272] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for IoT," in *Proc. ACM 2nd Int. Conf. Internet Things Design Implement.*, 2017, pp. 173–178.

[273] M. Carlsten, H. A. Kalodner, S. M. Weinberg, and A. Narayanan, "On the instability of bitcoin without the block reward," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 154–167. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978408

[274] S. Leonardos, D. Reijsbergen, and G. Piliouras, "PREStO: A systematic framework for blockchain consensus protocols," *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1028–1044, Nov. 2020.

[275] K. Wüst and A. Gervais, "Do you need a blockchain?" in *Proc. Crypto Valley Conf. Blockchain Technol. (CVCBT)*, 2018, pp. 45–54.

[276] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm, "What's inside the cloud? An architectural map of the cloud landscape," in *Proc. ICSE Workshop Softw. Eng. Challenges Cloud Comput.*, 2009, pp. 23–31.

[277] J. Verschuren, R. Govaerts, and J. Vandewalle, "ISO-OSI security architecture," in *Computer Security and Industrial Cryptography*. Vancouver, BC, Canada: Springer, 1993, pp. 179–192.

[278] S. M. Bellovin, "Security problems in the TCP/IP protocol suite," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 2, pp. 32–48, 1989.

[279] P. Pritzker and W. E. May, *Secure Hash Standard (SHS)*, Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, 2015.

[280] R. Cavanagh, "Digital signature standard; request for comments on the NIST-recommended elliptic curves," Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, Rep. 186-4, 2015.

[281] A. Narayanan. (2018). *Analyzing the 2013 Bitcoin Fork: Centralized Decision-Making Saved the Day*. [Online]. Available: https://freedom-to-tinker.com/2015/07/28/analyzing-the-2013-bitcoin-fork-centralized-decision-making-saved-the-day/

[282] A. Hertig. (2018). *Bitcoin Bug Exploited on Crypto Fork as Attacker Prints 235 Million Pigeoncoins*. [Online]. Available: https://www.coindesk.com/bitcoin-bug-exploited-on-crypto-fork-as-attacker-prints-235-million-pigeoncoins

[283] Bitcoin Wiki. (2016). *Value Overflow Incident*. [Online]. Available: https://en.bitcoin.it/wiki/Value_overflow_incident

[284] Ocminer. (2018). *Network Attack on XVG/VERGE*. Accessed: Jun. 9, 2019. [Online]. Available: https://bitcointalk.org/index.php?topic=3256693.0

[285] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.

[286] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where is current research on blockchain technology? A systematic review," *PLoS ONE*, vol. 11, no. 10, 2016, Art. no. e0163477.

[287] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Gener. Comput. Syst.*, vol. 107, pp. 841–853, Jun. 2020.

[288] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3416–3452, 4th Quart., 2018.

[289] M. Saad *et al.*, "Exploring the attack surface of blockchain: A systematic overview," 2019. [Online]. Available: arxiv.abs/1904.03487.

[290] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A survey on ethereum systems security: Vulnerabilities, attacks, and defenses," *ACM Comput. Surveys*, vol. 53, no. 3, pp. 1–43, 2020.

[291] D. Floyd. (2018). *150K Stolen From MyEtherWallet Users in DNS Server Hijacking*. [Online]. Available: https://www.coindesk.com/150k-stolen-myetherwallet-users-dns-server-hijacking

[292] L. Poinsignon. (2018). *BGP Leaks and Cryptocurrencies*. [Online]. Available: https://blog.cloudflare.com/bgp-leaks-and-crypto-currencies/

[293] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *ACM Comput. Surveys*, vol. 52, no. 3, p. 51, Jul. 2019. [Online]. Available: https://doi.org/10.1145/3316481

[294] A. Alkhalifah, A. Ng, A. Kayes, J. Chowdhury, M. Alazab, and P. Watters. (2019). *A Taxonomy of Blockchain Threats and Vulnerabilities*. [Online]. Available: https://doi.org/10.20944/preprints201909.0117.v1.

[295] L. Zhu *et al.*, "Research on the security of blockchain data: A survey," 2018. [Online]. Available: arxiv.abs/1812.02009.

[296] C. Natoli, J. Yu, V. Gramoli, and P. Veríssimo, "Deconstructing blockchains: A comprehensive survey on consensus, membership and structure," 2019. [Online]. Available: arxiv.abs/1908.08316.

[297] T. Neudecker and H. Hartenstein, "Network layer aspects of permissionless blockchains," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 838–857, 2nd Quart., 2018.

[298] D. Harz and W. Knottenbelt, "Towards safer smart contracts: A survey of languages and verification methods," 2018. [Online]. Available: arxiv.abs/1809.09805.

[299] M. Di Angelo and G. Salzer, "A survey of tools for analyzing Ethereum smart contracts," in *Proc. IEEE Int. Conf. Decentralized Appl. Infrastruct. (DAPPCON)*, 2019, pp. 69–78.

[300] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1432–1465, 2nd Quart., 2020.

[301] J. A. Garay and A. Kiayias, "SoK: A consensus taxonomy in the blockchain era," in *Proc. IACR Cryptol. ePrint Archive*, vol. 2018, 2018, p. 754.

[302] 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*, vol. 7, pp. 43622–43636, 2019.

[303] S. Azouvi and A. Hicks, "SoK: Tools for game theoretic models of security for cryptocurrencies," 2019. [Online]. Available: arxiv.abs/1905.08595.

[304] J. Huang *et al.*, "Survey on blockchain incentive mechanism," in *Proc. Int. Conf. Pioneering Comput. Sci. Eng. Educ.*, 2019, pp. 386–395.

[305] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito, "Blockchain and IoT integration: A systematic survey," *Sensors*, vol. 18, no. 8, p. 2575, 2018.

[306] 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 Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1676–1717, 3rd Quart., 2018.

[307] F. Restuccia, S. D'Oro, S. S. Kanhere, T. Melodia, and S. K. Das, "Blockchain for the Internet of Things: Present and future," 2019. [Online]. Available: arxiv.abs/1903.07448.

[308] 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 Things J.*, vol. 6, no. 2, pp. 2188–2204, Apr. 2019.

[309] Team Rocket. *Snowflake to Avalanche: A Novel Metastable Consensus Protocol Family for Cryptocurrencies*. 2018. [Online]. Available: https://www.semanticscholar.org/paper/Snowflake-to-Avalanche-%3A-A-Novel-Metastable-Family/85ec19594046bbcfe12137c7c2e3744677129820?p2df

[310] S. Popov. (2016). *The Tangle*. [Online]. Available: http://tanglereport.com/wp-content/uploads/2018/01/IOTA_Whitepaper.pdf

[311] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security*, 2001, pp. 253–280.

[312] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture," in *Proc. USENIX Security*, 2014, pp. 781–796.

[313] S. Micali, "Simple and fast optimistic protocols for fair electronic exchange," in *Proc. ACM 22nd Annu. Symp. Principles Distrib. Comput.*, 2003, pp. 12–19.

[314] Bitcoin Wiki. (2018). *Atomic Swap*. [Online]. Available: https://en.bitcoinwiki.org/wiki/Atomic_Swap

[315] D. Shares. (2016). *Gatecoin Official Statement: Hot Wallet Breach Losses Estimated to Be $2m USD*. [Online]. Available: https://news.bitcoin.com/gatecoin-official-statement-hot-wallet-breach-losses-estimated-2m-usd/

[316] P. Delaney. (2014). *Major Attack Reported on Dogecoin Vault Leads to Shutdown*. [Online]. Available: https://www.ccn.com/major-attack-reported-dogecoin-vault-leads-shutdown/

[317] S. Khandelwal. (2013). *Danish Bitcoin Exchange Bips Hacked and 1,295 Bitcoins Worth $1 Million Stolen*. [Online]. Available: https://thehackernews.com/2013/11/danish-bitcoin-exchange-bips-hacked-and_25.html

[318] S. Higgins. (2016). *The Bitfinex Bitcoin Hack: What We Know (and Don't Know)*. [Online]. Available: https://www.coindesk.com/bitfinex-bitcoin-hack-know-dont-know

[319] D. Bradbury. (2013). *Hackers Hit Bitcoin Central Exchange*. [Online]. Available: https://www.coindesk.com/hackers-hit-bitcoin-central-exchange

[320] C. Jentzsch. (2016). *The History of the DAO and Lessons Learned*. [Online]. Available: https://blog.slock.it/the-history-of-the-dao-and-lessons-learned-d06740f8cfa5/

[321] K. Elby. (2016). *King of the Ether Throne: Post Mortem Investigation*. [Online]. Available: https://www.kingoftheether.com/postmortem.html

[322] S. C. Bürgel. (2016). *Rock Paper Scissors on Ethereum*. [Online]. Available: https://github.com/SCBuergel/ethereum-rps

[323] H. Official. (2018). *Unit Underflows and Overflows—Ethereum Solidity Vulnerability*. [Online]. Available: https://medium.com/haloblock/unit-underflows-and-overflows-ethereum-solidity-vulnerability-39a39355c422

[324] B. Mahendar. (2019). *Unexpected Ether*. [Online]. Available: https://www.nvestlabs.com/2019/04/23/unexpected-ether/

[325] P. Technologies. (2017). *The Multi-Sig Hack: A Postmortem* [Online]. Available: https://www.parity.io/the-multi-sig-hack-a-postmortem/

[326] E. Ward. (2018). *Breaking Randomness in the Ethereum Universe*. [Online]. Available: https://blog.gdssecurity.com/labs/2018/6/1/breaking-randomness-in-the-ethereum-universe-part-1.html

[327] F. Hildebrand. (2018). *The State of Ponzi Schemes*. [Online]. Available: https://medium.com/solidified/the-state-of-ponzi-schemes-98dde4518431

[328] J. Wilcke. (2016). *The Ethereum Network Is Currently Undergoing a DoS Attack*. [Online]. Available: https://blog.ethereum.org/2016/09/22/ethereum-network-currently-undergoing-dos-attack/

[329] GuardRails. *Write to Arbitrary Storage Locations*. Accessed: Aug. 16, 2019. [Online]. Available: https://www.guardrails.io/docs/en/vulnerabilities/solidity/write_to_arbitrary_storage_location

[330] A. Akentiev. (2017). *Parity Multisig Hacked. Again*. [Online]. Available: https://medium.com/chain-cloud-company-blog/parity-multisig-hack-again-b46771eaa838

[331] A. Butler. (2019). *Ethereum Classic Attacked! How Does the 51% Attack Occur?*. [Online]. Available: https://hackernoon.com/ethereum-classic-attacked-how-does-the-51-attack-occur-a5f3fa5d852e

[332] D. Gutteridge. (2018). *Japanese Cryptocurrency Monacoin Hit by Selfish Mining Attack*. [Online]. Available: https://www.ccn.com/japanese-cryptocurrency-monacoin-hit-by-selfish-mining-attack/

[333] G. Milton. (2018). *Hackers Launch 'Double-Spend' Attack on Bitcoin Gold to Steal Over $18 Million*. [Online]. Available: https://cyware.com/news/hackers-launch-double-spend-attack-on-bitcoin-gold-to-steal-over-18-million-011a29e2

[334] Bitcoin Exchange Guide News Team. (2018). *Litecoin Cash (LCC) Experiencs 51% Attack, Cryptocurrency Dies on Spot*. [Online]. Available: https://bitcoinexchangeguide.com/litecoin-cash-lcc-experiencs-51-attack-cryptocurrency-dies-on-spot/

[335] ZenCash. (2018). *Zencash Statement on Double Spend Attack*. [Online]. Available: https://blog.zencash.com/zencash-statement-on-double-spend-attack/

[336] K. Sedgwick. (2018). *Verge Is Forced to Fork After Suffering a 51% Attack*. [Online]. Available: https://news.bitcoin.com/verge-is-forced-to-fork-after-suffering-a-51-attack/

[337] A. Judmayer *et al.*, "Pay-to-win: Incentive attacks on proof-of-work cryptocurrencies," *Proc. IACR Cryptol. ePrint Archive*, 2019.

[338] A. Greenberg. (2014). *Hacker Redirects Traffic From 19 Internet Providers to Steal Bitcoins*. [Online]. Available: https://www.wired.com/2014/08/isp-bitcoin-theft/

[339] D. Canellis. (2019). *Bitcoin Wallet Electrum Hit by DoS Attack From 140,000-Strong Botnet*. [Online]. Available: https://thenextweb.com/hardfork/2019/04/08/bitcoin-wallet-electrum-dos-attack-botnet-phishing/

[340] M. Apostolaki *et al.* (2017). *Blockchain Meets Internet Routing*. [Online]. Available: https://btc-hijack.ethz.ch/

[341] M. Saad, L. Njilla, C. Kamhoua, J. Kim, D. Nyang, and A. Mohaisen, "Mempool optimization for defending against ddos attacks in pow-based blockchain systems," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, 2019, pp. 285–292.

[342] J. Young. (2017). *Analyst: Suspicious Bitcoin Mempool Activity, Transaction Fees Spike to $16*. [Online]. Available: https://cointelegraph.com/news/analyst-suspicious-bitcoin-mempool-activity-transaction-fees-spike-to-16