A Security Reference Architecture for Blockchains

Ivan Homoliak* ihomoliak@sutd.edu.sg
Sarad Venugopalan* sarad_venugopalan@sutd.edu.sg
Qingze Hum* qingze_hum@mymail.sutd.edu.sg
Pawel Szalachowski* pawel@sutd.edu.sg

*Singapore University of Technology and Design

***Abstract*—Due to their interesting features, blockchains have become popular in recent years. They are full-stack systems where security is a critical factor for their success. The main focus of this work is to systematize knowledge about security and privacy issues of blockchains. To this end, we propose a security reference architecture based on models that demonstrate the stacked hierarchy of various threats (similar to the ISO/OSI hierarchy) as well as threat-risk assessment using ISO/IEC 15408. In contrast to the previous surveys [39], [8], [139], [20], we focus on the categorization of security incidents based on their origins and using the proposed architecture we present existing prevention and mitigation techniques. The scope of our work mainly covers aspects related to decentralized nature of blockchains, while we mention common operational security issues and countermeasures only tangentially.**

***Index Terms*—blockchain • distributed ledgers • reference architecture • threat-risk assessment**

I. 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 by untrusting participants 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.

Involved Parties. Blockchains usually involve the following parties (see Fig. 1).

(1) Consensus nodes actively participate in the underlying consensus protocol. These nodes can read the blockchain and write to it by appending new transactions. Besides, they can validate the blockchain and thus check whether writes of other consensus nodes are correct and respect a specified logic. By a combination of writing and validation capabilities, consensus nodes can prevent malicious behavior (e.g., by not appending invalid transactions, or not following an incorrect blockchain view). These nodes disseminate transactions to be appended within a block to the blockchain. In the context of Proof-of-Resource protocols (see Sec. IV-B), these nodes are often referred to as *miners*. *(2) Validating nodes* read the entire blockchain, validate it, and disseminate transactions to be appended to the blockchain. Unlike consensus nodes, validating nodes cannot write to the blockchain. Thus, they cannot prevent malicious behavior. However, since they possess copies of the entire blockchain, they can detect malicious behavior. *(3) Lightweight nodes* (a.k.a., clients) benefit from most of the blockchain functionalities, but they are equipped only with limited information about the blockchain. These nodes read only a fragment of the blockchain (usually block headers) and validate only a small number of transactions that concern them, while they rely on consensus and validating nodes for ensuring correctness of the blockchain. Therefore, they can detect only a limited set of attacks, usually pertaining to their own transactions.

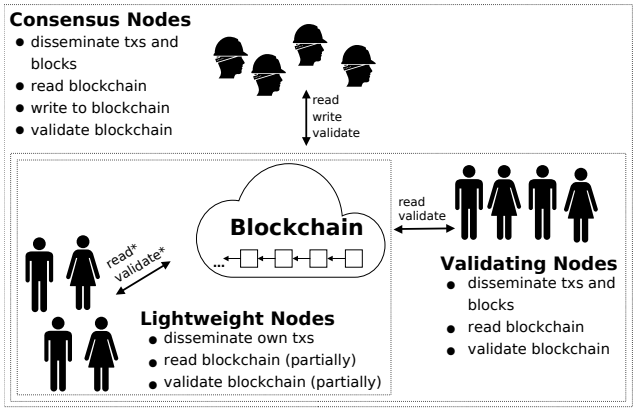

Fig. 1: Involved parties with their interactions and hierarchy.

Features of Blockchains. Blockchains were initially proposed as open cryptocurrencies, but due to their features, they became appealing for other applications as well. Blockchains achieve *decentralization* via a distributed consensus protocol, which provides resilience to failures. Usually, participants are equal and no single entity pose an authority. Another important result of decentralization is censorship resistance. The ledger is *immutable*, requiring a significant quorum of colluding nodes to change its entries retrospectively. Usually, immutability is achieved thanks to a cryptographic one-way function that creates integrity preserving links between blocks. Although blockchains are highly redundant in a storage of the data, the main advantage of such redundancy is high *availability*. This feature is of special interest to applications that cannot tolerate outages. Blockchain transactions, as well as actions of protocol

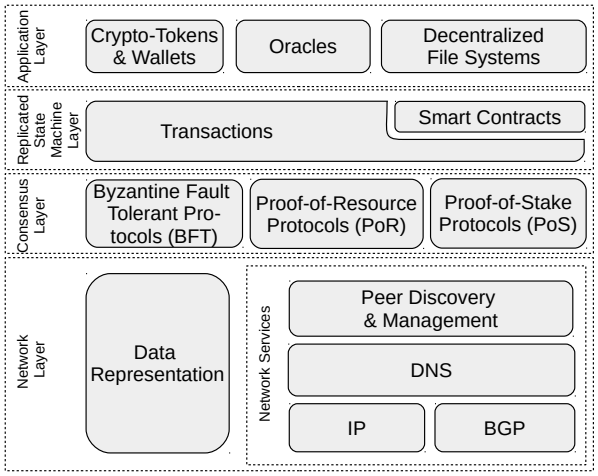

Fig. 2: Stacked model of reference architecture.

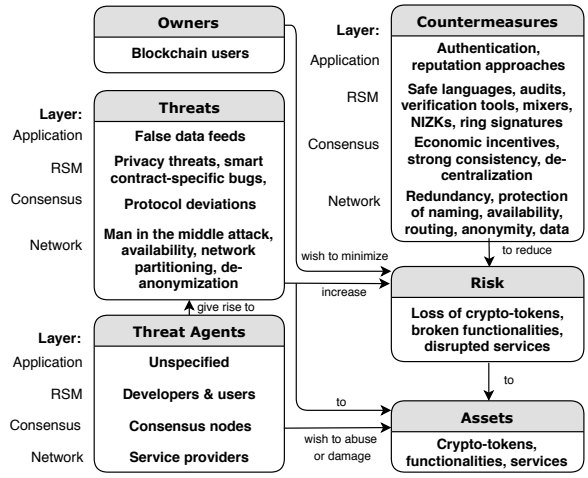

Fig. 3: Threat-risk assessment model of reference architecture.

participants, are usually *transparent* to other participants and in most cases even to the public. This can be a benefit for multiple applications, but it can also be seen as a disadvantage from the anonymity and privacy perspective.

Beside the features that are common in blockchains, some blockchains may focus on additional features, such as energy efficiency [60], [10], [75], scalability [95], throughput [25], [77], [144], privacy [121], accountability [72], etc.

Types of Blockchains. Based on how a new node enters a consensus protocol, we distinguish the following blockchain types. *(1) Permissionless* blockchains allow anyone to join the consensus protocol without permission. Such participation can be anonymous, and these protocols are designed to run over the Internet. To prevent Sybil attacks [45], these schemes usually require consensus nodes to establish their identities by running a Proof-of-Resource scheme, while the consensus power of a node is proportional to its resources invested into running the protocol. *(2) Permissioned* blockchains require a consensus node to obtain permission (and identity) to join the consensus protocol. The permission is given by a centralized or federated authority(ies), while nodes usually have equal consensus power (i.e., one vote per node). These schemes can be *public* if they are accessible over the Internet or *private* when they are deployed over a restricted network. *(3) Semi-Permissionless* blockchains require each new-coming consensus node to obtain a permission (i.e., cryptocurrency "stake"); however, such permission can be given by any stakeholder (i.e., consensus node). These blockchains are similar to permissionless blockchains, except a consensus is based on a stake rather than on resources spent. The node's consensus power is proportional to the stake it has. Similar to permissionless blockchains, these systems are usually intended to be run over the Internet. Novel and interesting aspects of (semi-)permissionless blockchains are incentives and network effects that are designed to increase the protocol's security, deployability, and adoption.

II. Security Reference Architecture

Stacked Model. To classify security aspects related to blockchains, we introduce a simplified stacked model [139] consisting of four layers (see Fig. 2). In contrast to previous work [139], we preserve only such granularity level that enables us to isolate various nature of security threats.

(1) The *network layer* (see Sec. III) consists of the data representation and network services planes. The data representation plane deals with storing and encoding of data, while the network service plane contains discovery and communication with protocol peers, addressing, routing, and naming services. (2) The *consensus layer* (see Sec. IV) deals with ordering of transactions and we divide it according to a type of the protocol used to Byzantine Fault Tolerant (e.g., [34], [13], [29], [47], [92]), Proof-of-Resource (e.g., [95], [48], [99], [89], [108]), and Proof-of-Stake (e.g., [10], [75]) protocols. (3) The *replicated state machine (RSM) layer* (see Sec. V) deals with the interpretation of transactions, according to which, the state of the blockchain is updated. Smart contracts involve two special types of transactions, which represent a programming code itself and invocations of this code together with input data. (4) In the application layer (see Sec. VI) we present the most common end-user functionalities such as crypto-tokens with wallets that store secrets, oracles that represent data feeds, and decentralized file systems. Throughout the paper, we summarize components of particular layers with their respective security threats and protection techniques.

Threat-Risk Assessment Model. To better capture security-related aspects of blockchain systems, we introduce a threat-risk model based on the template of ISO/IEC 15408 [36]. The model includes the following components and actors (see Fig. 3). *Owners* are blockchain users who run any node type. Owners posses crypto-tokens and/or use blockchain-based applications or services. *Assets* consist of monetary value (i.e., crypto-tokens), blockchain functionalities, as well as services built on top of them (e.g., exchanges, secure logging, supply chains). *Threat agents* are malicious users whose intention is to steal assets, break functionalities, or disrupt services. *Threats* arise from vulnerabilities at the network, in smart contracts, from consensus protocol deviations, violations of protocol assumptions, or application-specific dependencies. Threats facilitate various attacks on assets and services. *Coun-*

termeasures are provided by the security, safety, incentives, and reputation techniques that protect owners from threats. *Risks* caused by threats and their agents may lead to losses of monetary assets or service malfunctions and disruptions.

The owners wish to minimize the risk caused by threats that arise from threat agents. With the 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 come 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, nodes may be malicious and wish to alter the outcome of the consensus protocol by deviating from it. The countermeasures include economic incentives, strong consistency, and decentralization. At the RSM layer, the threat agents may stand for developers who (un)intentionally introduce semantic bugs in smart contracts (intentional bugs represent backdoors).[1] Mitigating countermeasures are safe languages, static/dynamic verification, and audits. Other threats are related to privacy of data and identity of users with mitigation techniques using mixers, privacy-preserving cryptography constructs (e.g., non-interactive zero-knowledge proofs (NIZKs), ring signatures). At the application layer, threat agents are unspecified, since any user on the network who uses a blockchain application may pose a threat. The threats on this layer arise from false data feeds and examples of mitigation techniques are authentication or reputation systems.

III. Network Layer

Blockchains are *overlay networks* on top of other networks; hence, blockchains inherit security and privacy issues from the underlying networks. Based on permission to join the blockchain system, the networks are either private or public. A private network is a network of local devices whose access is insulated from public networks. The Internet is a public network of interconnected autonomous systems (ASes) that relay network traffic at their borders. The network layer is divided into data representation and network sub-planes (see Fig. 2). 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 blockchain-oriented services provided by the network layer are peer management and discovery, which rely on the internals of underlying network, such as domain name resolution (i.e., DNS), network routing protocols (e.g., LAN routing for IP, WAN routing such as BGP). In the following, we discuss pros and cons of private and public networks and their security threats that affect overlayed blockchains.

[1]Note that semantic bugs may occur at each layer; however, we focus only on smart contract-specific bugs that we put into the RSM layer.

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. The inherent feature of private networks is that authentication and access control can be provided at the network layer.

1) Pros: Access control is achieved by centralized authentication of users and assigning them roles. A private network has full control over routing paths and physical resources used, which enables regulation of the network topology and transmission medium best suited for requirements. *Data privacy* is ensured by permissioned settings. *User identity* is revealed only within a private group of nodes. They are immune to external attacks in contrast to public networks (see Sec. III-B3). *Fine-grained authorization controls* are applied by the operator of 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 service over the Internet. *Applicability* of private networks is suitable only for permissioned and private blockchains.

3) Security Threats and Countermeasures: Insiders may pose a serious threat to security [5]. A compromised node may already have administrative privileges or obtain them by exploiting a system, network, or security vulnerabilities. *Countermeasures* include regular software updates, user monitoring (e.g., SIEM [130]), prevention techniques that minimize trust and maximize trustworthiness, as well as respecting best practices [70].

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 find useful to disseminate 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 [115]. *Openness and low entry barrier* 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

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

regulations) factors [23]. Statistical resource sharing [87] and openness are fundamental to a low entry barrier.

2) Cons: Single-point-of-failure – DNS with its hierarchy, IP addresses, and ASes are managed by centralized parties – Internet Corporation for Assigned Names and Numbers (ICANN); in particular, Internet Assigned Numbers Authority (IANA). *External adversaries* pose a threat to public networks. These adversaries can be classified based on their capabilities to which the blockchain network may be exposed [117]: (1) resources under attacker control (e.g., botnets, DNS and BGP servers), (2) identities are stolen or masqueraded (e.g., IP addresses participating in an eclipse attack or route manipulation), (3) MITM attacker (i.e., eavesdropping and spoofing), (4) common vulnerabilities leading to exploits (e.g., observed in DNS BIND software [52]), (5) revealing secrets (e.g., deanonymizing peers). *Efficiency* – although an average Internet bandwidth has improved in recent years [1], a distribution of powerful infrastructure is not uniform, which results in a different latency among peers, and the overall latency of the network is increased – this, in turn, may result to loss of created blocks and thus wasting of consensus power.

3) Security Threats and Countermeasures:

DNS attacks commonly arise from cache poisoning [128] that mainly affects nodes employing DNS bootstrapping [17] to retrieve online peers but also users of online blockchain explorers.[3] 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 [40].

Routing attacks are traffic route diversions, hijacking, or DoS attacks. Beside simple data eavesdropping or modification, these attacks may lead to network partitioning, which in turn raises the risks of 51% attacks or selfish mining attacks (see Sec. IV). *Countermeasures* suggest nodes to be multi-homed (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 [3]. Another mitigation is SABRE [2], a secure relay network that runs alongside with the Bitcoin network. The BGPsec [80] is a security extension for BGP used between neighboring ASes, and it provides assurance of route origin and propagation by cryptographic verification.

Eclipse attacks hijack all connections of a node to the blockchain network. Hence, all traffic received by the node is under the full control of the attacker. Eclipse attacks arise from threats on DNS and routing in the network as well as they may be a result of vulnerabilities in p2p protocols [66], [143], [85]). Eclipse attack increase chances of selfish mining and double spending attacks (see Sec. IV) – the eclipsed victims may vote for an attacker's chain. *Countermeasures:* Improving randomness in choosing peers was proposed in work [66] by several rules that manage the peer table. Another mitigation strategy against eclipse attacks is to use redundant network links or out-of-band connections to verify transactions (e.g., by a blockchain explorer). Also, note that countermeasures for DNS and routing attacks are applicable here as well.

DoS attacks on connectivity of consensus nodes may result in a loss of consensus power, thus preventing consensus nodes from being rewarded [67]. For validating nodes, this attack leads to disruption of some blockchain dependent services [124]. *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 [4] (i.e., combinations of the previous two).

DoS attacks on local resources, such as memory and storage, may reduce the peering and consensus capabilities [81] 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. Possible mitigation is raising the minimum transaction fee and rate-limit to the number of transactions. Several mitigating techniques are applied to Bitcoin [18] nodes including scoring DoS attacks and banning misbehaving peers.

Identity revealing attacks are conducted by linking the IP address of a node with identity propagated in transactions [15], [91]. Traffic analysis using Sybil listeners can reveal the linkage of node IP addresses and their transactions [33]. *Countermeasures* include using VPNs or anonymization services, such as Tor. See Sec. V-A1 for further identity and privacy-protecting mechanisms at the RSM level.

IV. Consensus Layer

The consensus layer of the stacked model deals with the ordering of transactions. It includes three main categories of consensus protocols with regard to different principles of operation and thus their security aspects. First, we focus on the security aspects that are generic to all categories of consensus protocols, and then we detail each category.

A. Generic Attacks

1) Violations of Protocol Assumptions:

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

[3]Note that blockchain explorers might be also affected by compromised certification authorities. Protection relying on DNS is DNS-based Authentication of Named Entities (DANE), while Certificate Transparency is mitigation relaying on centralized public logs.

reward honest participation and discourage [90] or punish [28], [41] protocol violations.

Time-Validation 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 [22]. 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 [132].

2) Double Spending: 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. To prevent this attack in permissionless blockchains, it is recommended to wait a certain amount of time (i.e., a few next blocks) until a block "is settled."

B. Proof-of-Resource Protocols (PoR)

Protocols from this category require nodes to prove a spending of a scarce resource in a lottery-based fashion [72]. 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 [48] (e.g., Spacecoin [102], SpaceMint [71]). *(3) Crypto-tokens* spent for Proof-of-Burn protocols (e.g., Slimcoin [99]). *(4) Combinations and modification* of the previous types, such as storage and computation, called Proof-of-Retrievability (e.g., Permacoin [89]) and storage over time, which is represented by Proof-of-Space protocols (e.g., Filecoin [108]).

PoR protocols belong to the first generation of consensus protocols, and they are mostly based on Nakamoto Consensus [95] that utilize PoW, inheriting its pros (e.g., high scalability) and cons (e.g., low throughput). For the detailed analysis of several PoW designs, we refer the reader to [149].

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

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

3) Security Threats and Mitigations:

Selfish Mining: In selfish mining [56], an adversary attempts to privately build a secret chain and reveal it to the public only when an honest chain is "catching up" with the secret one. The longest chain rule causes honest miners to adopt the attacker's chain and invalidate the honest chain, thus wasting their consensus power. This attack is more efficient when consensus power of a selfish miner reaches some threshold (e.g., 30%). The selfish mining strategy was later generalized [120] and extended to other variants that increase the profit of the attacker [96]. *Countermeasures:* (1) For the case of the longest chain rule, the first introduced mitigation is uniform tie breaking [56], 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 [120]. (2) As the longest chain rule enables this attack, it is recommended to use other fork choice rules that also account for the quality of solutions and make the decision deterministic, as opposed to a uniform tie breaking. An example of such a rule is to select the block based on the smallest hash value. Another example is to include partial solutions [145], [103] or full (orphaned) blocks [127], [148] for computation of block's quality. (3) Another option for a deterministic fork choice rule is using a pseudo-random function [76], which moreover provides unpredictability, hence an attacker cannot determine his chances to win a tie. (4) PoW protocols can be combined with BFT protocols, where PoW is used only for joining the protocol and BFT for consensus itself (e.g., [77], [144], [76]).

Feather Forking: In this attack [88], 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 honest nodes prefer to join on 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 [145], [114], [103] or full orphaned blocks [127], [148] into branch difficulty computation.

Pool Specific Attacks: Since PoR protocols are usually based on a lottery having a single winner [95], rewarding for participation imposes a high payout variance for solo miners (i.e., once in a few years). As a consequence, mining pools emerged and caused centralization of the mining power, which may result in selfish mining, double spending, or 51% attacks. *Countermeasures:* Non-outsourceable scratch-off puzzles [90] avoid creation of pools but require each consensus node to meet high demands on connectivity and storage, as opposed to centralized pools, where only a pool operator needs to meet these demands. If pools are acceptable, their size can be controlled by protocols that reward partial solutions [145], [114], [103] and thus minimize payout variance. For a detailed analysis of rewarding schemes in pools, we refer the reader to [116].

C. Byzantine Fault Tolerant (BFT) Voting Protocols

BFT protocols represent voting-based [72] consensus protocols that utilize Byzantine agreement and a state machine replication [122]. These protocols assume a fully connected topology, broadcasting messages, and a master-

replicas hierarchy. Synchronous examples of this category are PBFT [34], RBFT [7], eventually synchronous examples are BFT-SMaRt [13], Tendermint [25], Byzantine Paxos [29], BChain [47], and asynchronous examples are SINTRA [31] and HoneyBadgerBFT [92]. For more details, we refer the reader to review of BFT protocols and their practical applications in both permissioned and permissionless blockchains [32].

1) Pros: BFT protocols provide high throughput and a low latency finality. To face their scalability limitation, BFT protocols are often combined with PoS or PoW. This is in line with a lottery approach [72] for selecting a portion of all nodes, referred to as committee, which further runs BFT consensus (e.g., Algorand [60], Zilliqa [152], DFINITY [64]).

2) Cons: The main con of traditional BFT protocols [29], [34] is a low scalability caused by a high communication complexity (i.e., $\Theta(n^2)$). Since these protocols can work efficiently only with a limited number of consensus nodes, they can be used in their pure form only in permissioned blockchains.

3) Security Threats and Mitigations: Many BFT protocols assume synchronous delivery of messages. However, this assumption can be violated by unpredictable network scheduler, as demonstrated on PBFT protocol [92]. This fact motivates asynchronous BFT protocols that can be based on threshold-based cryptography, which enables reliable and consistent broadcast [31], [92]. Issues with scalability and throughput can be dealt with by applying cryptographic constructs [19], [30], [125] and partitioning consensus nodes into shards that process transactions in parallel [77], [144]. Another option is to prune the number of nodes running BFT into committees [60], which, however, reduces security level of BFT and provides only probabilistic security guarantees depending on the committee size.

D. Proof-of-Stake Protocols (PoS)

Similar to the PoR category, PoS protocols are based on the lottery approach [72]. However, in contrast to PoR, no scarce resource is spent; instead, the nodes are required "to prove investment" of crypto-tokens in order to participate in a protocol, and thus potentially earn interest from the invested amount. The concept of PoS was first time proposed in Peercoin [119] as a combination with PoW – each node has its particular difficulty for PoW, which is based on the age of the coins a node owns. Although there exist a few pure PoS protocols (e.g., Chains of Activity [10], Ouroboros [75]), the trend is to combine them in a hybrid setting with PoR (e.g., Proof-of-Activity [11], Peercoin [119], Snow White [12]) or BFT protocols (e.g., Algorand [60]). In particular, a combination of PoS with BFT represents a promising approach, which takes advantages of both lottery and voting (i.e., scalability and throughput), while no resources are wasted.

1) Pros: The main feature of PoS protocols, as compared to PoR, is their energy efficiency. Although some PoS protocols are often combined with a PoR technique (e.g., [12], [119]), the overall energy spent is much less than in the case of pure PoR protocols.

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

3) Security Threats and Mitigations:

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 a chance to be rewarded. Such behavior increases the number of forks and thus time to finality. *Countermeasures:* Deposit-based solutions (e.g., [28]) require nodes to make a deposit during some fixed period/round and checkpoint-based solutions (e.g., [28], [24], [41]) 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 use cryptographic solutions [83] for revealing identity and a private key of a node that signs two conflicting blocks. Another countermeasure is to use backward penalization of nodes that produced two or more conflicting chains [41], [28]. Finally, PoS protocols can be combined with BFT approaches, and thus forks can hardly occur (e.g., [60]).

Grinding Attack: If the leader or committee producing a block is determined before the round starts, then the attacker can bias this process to increase his chances of being selected in 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:* A grinding attack can be prevented by performing a fresh leader election by an interaction of nodes (e.g., the secure multiparty coin flipping protocol [75]) or by private checking whether the output of a verifiable random function (VRF) is below a certain stake-specific threshold (e.g., [60]). The input of the VRF is the user's private key and the randomness unambiguously bound to the previous block; hence each consensus node computes the only VRF output during each round.

Denial of Service on a Leader/Committee: If a leader or a committee are publicly determined before the round starts [75], then the adversary may conduct a DoS attack against them and thus cause a restart of the round – this might be repeated until adversary's desired nodes are elected. *Countermeasures:* A prevention technique was proposed in Algorand [60] – a node privately determines whether it is a potential leader (or committee member), and immediately releases a block candidate (or a vote) – hence, after publishing this data, it is too late for a DoS attack. The concept of VRF approach was also utilized in other protocols (e.g., [42], [64]).

Long-Range Attack: In this attack [26] (a.k.a., posterior corruption [41]), 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 impose 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 vari-

ant of long-range attack that considers transaction fee-based rewarding and infrequent or no check-points is denoted as a *stake-bleeding* attack [59]. *Countermeasures:* One mitigation is to lock the deposit for a longer time than the period of participation in the consensus [8]. The next mitigation technique is frequent periodic check-pointing, which causes the irreversibility of the blockchain with respect to the last checkpoint. Another option is to apply key-evolving cryptography [58] and forward-secure digital signatures [9], which require users to evolve their private keys, while already used keys are erased [42]. Hence, signatures cannot be forged in the case of compromise. The third mitigation technique is enforcing a chain density in a time-domain [59] for the protocols where the expected number of participants in each round is known (e.g., [75]). The last mitigation technique is context-sensitive transactions, which put the hash of a recent valid block into a transaction itself [59].

V. Replicated State Machine Layer

This layer is responsible for the interpretation of transactions and concerning security threats are related to the privacy of users, confidentiality of data, and smart contract-specific bugs (involving bugs in code and compilers).

A. Transaction Protection

Mostly, transactions containing plain-text data are digitally signed by private keys of users [95], [54], enabling anybody to verify the validity of transactions by corresponding public keys. However, such an approach provides only pseudonymous identities that can be traced to real identities, and moreover, it does not ensure confidentiality of data [57].

1) Security Threats and Countermeasures:

Privacy Threats to User Identity. In most of the blockchains, user identities can be linked with their transactions by various deanonymization techniques, such as network flow analysis, address clustering, transaction fingerprinting [57], [14], [110]. Moreover, blockchains designed with anonymity and privacy features (e.g., Zcash, Monero) are also vulnerable to a few attack strategies [74], [94]. *Countermeasures:* Various means are used for obfuscation of user identities, including centralized [86], [21] and decentralized [118], [16], [151] mixing services, ring signatures [138], [98], and non-interactive zero-knowledge proofs (NIZKs) [121]. Some mixers enable internal linkability by involved parties [86] or linkability by the mixers [21], which are also potential threats. Unlinkability for all parties can be achieved by multi-party computation [151], blinding signatures [137], or layered encryption [118]. Ring signatures [113] provide unlinkability to users in a signing group [98], [138], enabling only verification of correctness of a signature, without revealing an identity of a signer.

Privacy of data. NIZKs [121], [51] and blind signatures [65], [137] can be used for preservation of data privacy. Another method is homomorphic encryption [100], [104], which enables to compute some operations over encrypted messages. Privacy and confidentiality for smart contract platforms can be achieved through trusted transaction managers [78], trusted hardware [35], and secure multi-party computations [153].

B. Smart Contracts

Smart contracts, introduced to automate legal contracts [131], 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 and contain arbitrary programming logic or only serve for limited purposes. In the following, we describe these two contrasting types of smart contract languages.

1) Security Threats and Countermeasures:

Turing-Complete Languages. An important aspect of this language category is a large attack surface due to the possibility of arbitrary programming logic. Examples of this category are Serpent and Solidity, while Solidity is the most popular and widely-used one. *Serpent* [55] is a high-level language that was designed to be simple and similar to the Python language. However, Serpent was designed in untyped fashion, lacking out-of-bound access checks of arrays and accepting invalid code by compilers [146], which opened the door for plenty of vulnerabilities. Hence, Serpent showed to be as an unsuccessful attempt to simplify the coding phase. *Solidity* [53] is an object-oriented statically-typed language that is primarily used by Ethereum platform. Contracts written in Solidity can contain various types of vulnerabilities [6], [84], which resulted in many incidents in the past. Mitigations of such vulnerabilities can be done by code analysis tools [101], [133], respecting best practices [126], [134], utilizing known design patterns [141], audits, and testing. Various approaches are used for source code analysis, such as linters [133], [109], [46], fuzzers [73], semantic-based program verifiers [68], and other symbolic code analyzers [135] often using control flow-graph techniques. Note that source code of contracts is often not public in contrast to their bytecode. For this reason, bytecode decompilers [150], [129], analyzers [97], and automated exploit generators [79] can be utilized.

Turing-Incomplete Languages. The main pro of this category is its design-oriented goal of small attack surface and emphasis on safety but at the cost of limited expressiveness. Examples of this category are Pact, Scilla, Vyper. *Pact* [107] is a declarative language intended for 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 SMT solvers. *Scilla* [123] is designed to achieve expressiveness and tractability while enabling formal reasoning about contract behavior. Every computation utilizes the automata-based model, and computations are realized as standalone atomic transitions that strictly terminate. Scilla enables external calls only as the last instruction of a contract, which simplifies proving safety and thus mitigates a few vulnerabilities. *Vyper* [27] is an experimental language designed to ease the audit of smart contracts and increase security – it contains strong typing and bounds/overflows checks.

VI. Application Layer

The application layer contains end-user services and applications that are built on top of blockchains; therefore the security threats are specific to particular types of applications. In the following, we elaborate on common application types.

A. Crypto-Tokens & Wallets

Besides cryptocurrencies that provide native crypto-tokens, there are other blockchain applications using crypto-tokens for the purpose of providing owners with rights against the third party (i.e., counterparty tokens) or with a possibility of transferring asset ownership (i.e., ownership tokens) [93]. All types of tokens require the protection of private keys and secrets linked with user identities. For this purpose, two main categories of wallets have emerged – *self-sovereign wallets* and *hosted wallets* [50], [20], [69]. Beside technical risks, all crypto-tokens are exposed to regulatory risk, while non-native tokens are in addition exposed to legal risks [93].

Self-Sovereign Wallets. Users of self-sovereign wallets locally store their private keys and directly interact with the blockchain platform using the keys to sign transactions. The instances of these wallets differ in several aspects. One of them is isolation of the keys – there are software wallets that store the keys within the user PC (e.g., Bitcoin Core, Electrum Wallet, MyEtherWallet) as well as hardware wallets that store keys in a sealed storage, while they expose only signing functionality (e.g., Trezor, Ledger, KeepKey, BitLox, CoolBitX). Another type of wallets enables to customize functionality and security by a smart contract (e.g., TrezorMultisig2of3 [136], Ethereum MultiSigWallet [38]).

Hosted Wallets. Hosted wallets require a centralized party to provide an interface for interaction with the wallet and thus blockchain. If a hosted wallet has full control over private keys, it is referred to as a *server-side wallet* (e.g., Coinbase, Circle Pay Wallet, Luno Wallet), while in the case of keys stored in the users' browsers, the wallets are referred to as *client-side wallets* (e.g., Blockchain Wallet, BTC Wallet, Mycelium Wallet, CarbonWallet, Citowise Wallet). We refer the reader to works [69], [50] for a security overview of miscellaneous wallet solutions.

1) Security Threats and Mitigations: Since server-side wallets accounted for several compromises [142], [111], [112], their popularity have attenuated in favor of client-side wallets. Client-side wallets do not expose private keys to a centralized party, but they still trust in the online interface provided by such a party, and moreover, their availability is dependent on such a party. Contrary, 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 [44], keyloggers [20], [106]). Possible mitigation of these attacks are hardware wallets displaying details of transactions to the user, while the user confirms signing by a button (e.g., Trezor, Ledger, KeepKey). Another option is to protect self-sovereign wallets by multi-factor/(-step) authentication using multi-signatures [136], [38], threshold-based cryptography [62], or air-gapped OTPs [69].

B. Oracles

Oracles are trusted entities that provide data reflecting the state of the world beyond the blockchain. *Prediction markets* (e.g., Augur [105], Gnosis [61]) were created for the purpose of trading the outcome of events – individuals are incentivized to accurately wager on these outcomes, which serve as data feeds. *Dedicated data feeds* build on existing blockchain platforms (e.g., PDFS [63], Oraclize [37]) or create dedicated oracle networks (e.g., ChainLink [49], Witnet [43]) that internally run consensus protocol.

1) Security Threats: The data provision time of prediction markets may be long for many applications and the provided set of data events may be also limited. In contrast, dedicated data feeds enrich a data domain and significantly shorten a provision time; however, they often rely on a trusted party [63], [37], which may misbehave or accidentally produce wrong data. Oracle networks eliminate trust in a single party by a consensus of the group; however, threats related to the consensus layer of this functionality also needs to be considered. Moreover, for providers that offer authenticated data feeds using trusted hardware [37], [147], a vulnerability in trusted hardware may result in a compromise of the entire data feed.

C. Decentralized Filesystems (DFs)

DFs serve as a data storage infrastructure running native blockchains (e.g., Storj [140], Filecoin [108], Permacoin [89]). DFs borrow ideas from peer-to-peer file storage systems, but they additionally incentivize data preservation by crypto-tokens. Alternatively to native DFs, decoupling of the stored data from the blockchain data is also possible in a few forms of integration with existing blockchains. Beside naïve storage of integrity proofs to off-chain data, cloud services (e.g., Amazon Web Services, Google Cloud, IBM), and distributed hash tables (DHT) [82] are promising approaches.

1) Security Threats and Mitigations: While native DFs handle availability and decentralization using consensus layer mechanisms, cloud services and DHT solutions rely on a provider's infrastructure and dedicated file sharing networks, respectively. Sybil attacks claiming redundant storage of the same piece of data can be prevented by unique encryption of each data copy [140], which, however, puts higher distribution overhead on clients. Another attack might target the reputation of the network by dropping data and its redundant copies. A simple mitigation technique is to use multiple consensus nodes for a file upload, which diminishes chances of the attack being successful. Next mitigation is to hide the number of redundant copies using erasure encoding [140].

VII. Conclusion

In this paper, we focused on the systematization of knowledge about security aspects of blockchain systems. We proposed security reference architecture as a stacked model, which we further projected into a threat-risk assessment model that presents various threats and countermeasures. The proposed stack model consists of four layers: (1) network layer,

(2) consensus layer, (3) replicated state machine layer, and (4) application layer. At each of the layers, we surveyed specific security issues and mitigation techniques. In future work, we plan to amend the security issues of each layer by details and evidence about real-world incidents.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation (NRF), Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2016NCR-NCR002-028) and administered by the National Cybersecurity R&D Directorate. Also, we would like to thank Pieter Hartel, Daniel Reijsbergen, and Stefanos Leonardos for their valuable feedback.

REFERENCES

[1] Akamai. Q1 2017 State of the Internet/Connectivity Report. Technical report, 2017.
[2] M. Apostolaki, G. Marti, J. Müller, and L. Vanbever. Sabre: Protecting bitcoin against routing attacks. 2018.
[3] M. Apostolaki, A. Zohar, and L. Vanbever. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *IEEE SP*, 2017.
[4] E. Arazi. Choosing the right ddos solution (part 4): Hybrid protection. https://blog.radware.com/security/2018/04/choosing-the-right-ddos-solution-hybrid-protection/, 2018.
[5] W. Ashford. Corporate networks vulnerable to insider attacks, report finds. https://www.computerweekly.com/news/252444419/Corporate-networks-vulnerable-to-insider-attacks-report-finds, 2018.
[6] N. Atzei, M. Bartoletti, and T. Cimoli. A survey of attacks on ethereum smart contracts (sok). In *POST*, 2017.
[7] P.-L. Aublin, S. B. Mokhtar, and V. Quéma. RBFT: Redundant byzantine fault tolerance. In *ICDCS*, 2013.
[8] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis. Consensus in the age of blockchains. *arXiv preprint arXiv:1711.03936*, 2017.
[9] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In *CRYPTO'99*, 1999.
[10] I. Bentov, A. Gabizon, and A. Mizrahi. Cryptocurrencies without proof of work. In *FC*, 2016.
[11] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake. 2014.
[12] I. Bentov, R. Pass, and E. Shi. Snow white: Provably secure proofs of stake. 2016.
[13] A. Bessani, J. Sousa, and E. E. Alchieri. State machine replication for the masses with bft-smart. In *IEEE/IFIP DSN*, 2014.
[14] A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *ACM CCS*, 2014.
[15] A. Biryukov and I. Pustogarov. Bitcoin over tor isn't a good idea. *CoRR*, abs/1410.6079, 2014.
[16] G. Bissias, A. P. Ozisik, B. N. Levine, and M. Liberatore. Sybil-resistant mixing for bitcoin. In *WPES*, New York, NY, USA, 2014.
[17] Bitcoinj team. Bitcoinj security model. https://bitcoinj.github.io/security-model, 2019.
[18] BitcoinWiki. Weaknesses. https://en.bitcoin.it/wiki/Weaknesses#Denial_of_Service_.28DoS.29_attacks, 24 July 2018.
[19] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, 2001.
[20] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *IEEE SP*, 2015.
[21] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *FC*, pages 486–504. Springer, 2014.
[22] A. Boverman. Timejacking & bitcoin, 2011.
[23] S. Box and J. K. West. Economic and social benefits of internet openness. https://ssrn.com/abstract=2800227, 22 June 2016.
[24] Bticoinwiki. Peercoin, 2019.
[25] E. Buchman, J. Kwon, and Z. Milosevic. The latest gossip on bft consensus. 2018.
[26] V. Buterin. Long-range attacks: The serious problem with adaptive proof of work. *Ethereum Blog, May*, 2014.
[27] V. Buterin. Vyper, 2017.
[28] V. Buterin and V. Griffith. Casper the friendly finality gadget. 2017.
[29] C. Cachin. Yet another visit to paxos. *IBM Research, Zurich, Switzerland, Tech. Rep. RZ3754*, 2009.
[30] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. *Journal of Cryptology*, 18(3), 2005.
[31] C. Cachin and J. A. Poritz. Secure intrusion-tolerant replication on the internet. In *DSN*, 2002.
[32] C. Cachin and M. Vukolić. Blockchain consensus protocols in the wild. 2017.
[33] G. Caffyn. Chainalysis ceo denies sybil attack on bitcoin network. https://www.coindesk.com/chainalysis-ceo-denies-launching-sybil-attack-on-bitcoin-network, 2015.
[34] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, 1999.
[35] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution. *arXiv preprint arXiv:1804.05141*, 2018.
[36] Common Criteria. Common criteria for information technologysecurity evaluation. part 1: Introduction and general model. Technical report, 2017.
[37] Concur Technologies, Inc. Oraclize documentation. https://github.com/kadena-io/pact/blob/ac759c0882d97b60473cfbb5853b1c25259e1213/docs/pact-properties.md, 2008.
[38] Consensys team. Ethereum MultiSigWallet, 2017.
[39] M. Conti, E. S. Kumar, C. Lal, and S. Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4), 2018.
[40] K. Crispin. Alt-roots, alt-tlds. IETF Draft, 2001.
[41] P. Daian, R. Pass, and E. Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proofs of stake. In *Iacr*. 2017.
[42] B. David, P. Gaži, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT*, 2018.
[43] A. S. de Pedro, D. Levi, and L. I. Cuende. Witnet: A decentralized oracle network protocol. 2017.
[44] Dell SecureWorks. Cryptocurrency-stealing malware landscape, 2015.
[45] J. R. Douceur. The sybil attack. In *IPTPS*, 2002.
[46] R. Dua. Solium documentation release 1.0.0. https://media.readthedocs.org/pdf/solium/latest/solium.pdf, 11 Feb 2019.
[47] S. Duan, H. Meling, S. Peisert, and H. Zhang. Bchain: Byzantine replication with high throughput and embedded reconfiguration. In *OPODIS*, 2014.
[48] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak. Proofs of space. In *CRYPTO'15*, 2015.
[49] S. Ellis, A. Juels, and S. Nazarov. Chainlink: A decentralized oracle network.
[50] S. Eskandari, J. Clark, D. Barrera, and E. Stobert. A first look at the usability of bitcoin key management. 2018.
[51] T. Espel, L. Katz, and G. Robin. Proposal for protocol on a quorum blockchain with zero knowledge. 2017, 2017.
[52] S. G. et al. Bind 9 security vulnerability matrix. https://kb.isc.org/docs/aa-00913, 2019.
[53] Ethereum team. Solidity. https://solidity.readthedocs.io/en/v0.5.4/index.html#.
[54] Ethereum team. A Next-Generation Smart Contract and Decentralized Application Platform. https://github.com/ethereum/wiki/wiki/White-Paper#modified-ghost-implementation, 2018.
[55] Etherum team. Serpent, 2017.
[56] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7), 2018.
[57] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar. A survey on privacy protection in blockchain system. *Journal of Network and Computer Applications*, 126, 2019.
[58] M. Franklin. A survey of key evolving cryptosystems. *International Journal of Security and Networks*, 1(1-2), 2006.
[59] P. Gaži, A. Kiayias, and A. Russell. Stake-bleeding attacks on proof-of-stake blockchains. In *IEEE CVCBT*, 2018.

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

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

[62] S. Goldfeder, R. Gennaro, H. Kalodner, J. Bonneau, J. A. Kroll, E. W. Felten, and A. Narayanan. Securing bitcoin wallets via a new dsa/ecdsa threshold signature scheme, 2015.

[63] J. Guarnizo and P. Szalachowski. Pdfs: Practical data feed service for smart contracts. 2018.

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

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

[66] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *USENIX Security*, 2015.

[67] S. Higgins. Bitcoin mining pools targeted in wave of ddos attacks. https://www.coindesk.com/bitcoin-mining-pools-ddos-attacks, 2015.

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

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

[70] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa. Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures. *ACM Computing Surveys (CSUR)*, 52(2):30, 2019.

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

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

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

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

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

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

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

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

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

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

[81] S. D. Lerner. New dos vuln by forcing continuous hard disk seek/read activity (fixed in 0.8.0). https://bitcointalk.org/index.php?topic=144122.0, 2019.

[82] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun. Blockchain for large-scale internet of things data storage and protection. *IEEE Transactions on Services Computing*, 2018.

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

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

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

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

[87] L. W. McKnight and J. P. Bailey. An introduction to internet economics. 1995.

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

[89] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz. Permacoin: Repurposing bitcoin work for data preservation. In *IEEE SP*, 2014.

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

[91] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee. Discovering bitcoin' s public topology and influential nodes. 2015.

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

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

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

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

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

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

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

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

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

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

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

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

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

[105] J. Peterson and J. Krug. Augur: a decentralized, open-source platform for prediction markets. 2015.

[106] A. Peyton. Cyren sounds siren over bitcoin siphon scam, 2017.

[107] S. Popejoy. The Pact smart contract language, 2016.

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

[109] Protofire. Solhint project. https://github.com/protofire/solhint, 2017.

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

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

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

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

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

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

[116] M. Rosenfeld. Analysis of Bitcoin pooled mining reward systems. 2011.

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

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

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

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

[121] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE SP*, 2014.

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

[123] I. Sergey, A. Kumar, and A. Hobor. Scilla: a smart contract intermediate-level language. 2018.

[124] D. Shares. Major DDoS attacks hit bitcoin.com. https://news.bitcoin.com/ddos-attacks-bitcoin-com-uncensored-information/, 2017.

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

[126] SmartContractSecurity. Smart contract weakness classification registry. https://github.com/SmartContractSecurity/SWC-registry/, 2019.

[127] Y. Sompolinsky and A. Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. 2013(881), 2013.

[128] S. Son and V. Shmatikov. The hitchhiker's guide to dns cache poisoning. In S. Jajodia and J. Zhou, editors, *SecureComm*, Berlin, Heidelberg, 2010.

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

[130] D. Swift. A Practical Application of-SIM/SEM/SIEM Automating Threat Identification. https://www.sans.org/reading-room/whitepapers/logging/practical-application-sim-sem-siem-automating-threat-identification-1781, 2006).

[131] N. Szabo. The idea of smart contracts.

[132] P. Szalachowski. (short paper) towards more reliable bitcoin timestamps. In *IEEE CVCBT*, 2018.

[133] S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov. Smartcheck: Static analysis of ethereum smart contracts. In *WETSEB*, 2018.

[134] trailofbits. Awesome ethereum security. https://github.com/trailofbits/awesome-ethereum-security, 11 Aug 2018.

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

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

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

[138] N. van Saberhagen. Cryptonote v 2.0. https://cryptonote.org/whitepaper.pdf, 2013.

[139] W. Wang, D. T. Hoang, Z. Xiong, D. Niyato, P. Wang, P. Hu, and Y. Wen. A survey on consensus mechanisms and mining management in blockchain networks. 2018.

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

[141] M. Wohrer and U. Zdun. Smart contracts: Security patterns in the ethereum ecosystem and solidity. In *IWBOSE*, 2018.

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

[143] K. Wüst and A. Gervais. Ethereum eclipse attacks. Technical report, 2016.

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

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

[146] Zeppelin Sollutions. Serpent compiler audit. https://docs.google.com/document/d/1_PqXuAkvgUAOG3jbBvaUvqN6W90eJ3N4IdTLNMRAijo/edit#heading=h.pe41jxc4c6xs, 2017.

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

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

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

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

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

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

[153] G. Zyskind, O. Nathan, and A. Pentland. Enigma: Decentralized computation platform with guaranteed privacy. 2015.