IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 12, 2020, accepted September 9, 2020, date of publication September 21, 2020, date of current version October 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3025622

BAD: A Blockchain Anomaly Detection Solution

MATTEO SIGNORINI[1], MATTEO PONTECORVI[1], WAËL KANOUN[2], AND ROBERTO DI PIETRO[3], (Senior Member, IEEE)
[1]NOKIA Bell Labs, 91620 Nozay, France
[2]Thales, Dubai, UAE
[3]College of Science and Engineering, ICT Division, Hamad Bin Khalifa University, Doha, Qatar

Corresponding author: Roberto Di Pietro (rdipietro@hbku.edu.qa)

Open Access funding provided by the Qatar National Library.

ABSTRACT Anomaly detection tools play a role of paramount importance in protecting networks and systems from unforeseen attacks, usually by automatically recognizing and filtering out anomalous activities. Over the years, different approaches have been designed, all focused on lowering the false positive rate. However, no proposal has addressed attacks specifically targeting blockchain-based systems. In this paper, we present BAD: Blockchain Anomaly Detection. This is the first solution, to the best of our knowledge, that is tailored to detect anomalies in blockchain-based systems. BAD is a complete framework, relying on several components leveraging, at its core, blockchain meta-data in order to collect potentially malicious activities. BAD enjoys some unique features: (i) it is distributed (thus avoiding any central point of failure); (ii) it is tamper-proof (making it impossible for a malicious software to remove or to alter its own traces); (iii) it is trusted (any behavioral data is collected and verified by the majority of the network); and, (iv) it is private (avoiding any third party to collect/analyze/store sensitive information). Our proposal is described in detail and validated via both experimental results and analysis, that highlight the quality and viability of our Blockchain Anomaly Detection solution.

INDEX TERMS Blockchain technology, security, intrusion detection systems, distributed systems.

I. INTRODUCTION

The *Internet of Things* (IoT) digital revolution has brought a wide range of smart devices in the global market that are remotely accessible via Internet and able to communicate and cooperate with each other. This opens great opportunities from an application and service point of view, but it also creates new security challenges as devices are easily accessible from Internet [1].

To address the above introduced typology of threat, *intrusion detection systems* (IDS) have been developed in the past as tools aimed at strengthening the security of complex networks and systems via capturing, monitoring, and analyzing the peers' traffic or, more in general, their behavior [2]. These approaches, usually based on log analysis and data correlation, aim at building attack models and mitigation strategies on top of them. Existing IDS can be classified based on their approach into two classes: signature recognition or anomaly behavior [3]. On the one hand, the first class leverages databases where signatures of well-known attacks are matched. These databases are then used as a reference model to detect future occurrences of such attacks. Hence, this approach is not able to recognize new attacks whose signatures are still unknown. On the other hand, anomaly detection approaches build models of normal behavior and rise alerts for deviations from such baselines. Thus, the goal of an *anomaly detection system* (ADS) is to build the normal behavior model and then to challenge it with new/unknown behaviors in order to analyze how close they are to the reference model.

IDS and *ADS* proved their functionalities so far, especially when based on trusted third parties that are responsible to build reference models and to alert end-users or end-devices if an unexpected behavior has been detected. We can consider the classic case of anti-virus companies that build and manage threat databases, which are later used to identify known threats or to predict zero-day attacks. However, this approach does not work for truly distributed peer-to-peer communities that lack trusted anchors or centralized management, as in blockchain-based applications. Firstly designed as a

The associate editor coordinating the review of this manuscript and approving it for publication was Marco Anisetti.

support tool for Bitcoin [4], the blockchain technology allows untrusted peers within open (i.e. permission-less) communities to agree on the status of a shared database, without the necessity to access trusted third parties. The only assumption is that the majority of involved peers is honest and willing to keep the protocol up and running against malicious users. However, has shown in real life applications, attackers can *eclipse* their victims (i.e. manipulate honest nodes access to the mainstream global blockchain), thus reducing the number of honest peers participating in the overall blockchain network. Eclipsing a node allows or simplifies several types of attacks as shown in [5], [6].

A. CONTRIBUTIONS

In this paper we propose BAD: a general solution that leverages the features of blockchain to provide an Anomaly Detection Service. As an instance of its effectiveness, BAD allows the peers in a blockchain network to be protected against eclipse attacks by sharing information on previous attacks (i.e. by re-distributing malicious forks to the whole peer-to-peer community). To the best of our knowledge, our approach is the first one that leverages forks on a global scale to detect and prevent local threats. The core idea behind BAD is to collect local attack logs in the form of (hashed) malicious transactions. These logs are generated by BAD from an attack sequence injected by an attacker on isolated victims, and they are later reused to prevent similar attacks on uncorrupted nodes. More precisely, the attack logs (usually discarded in standard blockchain applications) populate a threat database that allows other potential victims to be resilient against zero-day attacks already discovered. The proposed solution is detailed and implemented. Achieved results show its quality and viability, and pave the way for future research along the highlighted directions.

B. ROADMAP

This paper is organized as follows: in Section II the blockchain background technology is introduced as well as related previous works on anomaly detection systems. Section III describes our threat model. Sections IV and V introduce respectively our solution and the related experimental results. In Section VI we discuss the overhead analysis of BAD as well as its theoretical complexity, while Section VII addresses issues and limitations of our system. Finally, Section VIII concludes the paper and introduces future work.

II. TECHNOLOGY BACKGROUND AND RELATED WORK

In the rest of this paper we adopt the same notation used in [7] to describe blockchain and, in general, state machine replication protocols. We will only consider permissionless blockchain technologies, where a race among peers is established for mining blocks and rising potential forks (see Bitcoin, Ethereum and Tether which are the cryptocurrencies with higher market capitalization[1]). We give some concepts and definitions from [7] below, followed by a general description of a blockchain protocol.

An *output* is a tuple consisting of an amount of bitcoins and a spending condition. The latter is usually a valid signature associated with the private key of the spender address, however it can be generally a script which could be exploited by an attacker.

An *input* is a tuple consisting of a reference to a previously created output and arguments for the spending condition. This allows the transaction creator to spend the referenced output. We call *UTXO* the set of unspent transaction outputs.

Definition 1 [7]: A *transaction* is a data structure that describes the transfer of bitcoins from spender to recipients. The transaction consists of a number of inputs and new outputs. The former result in the referenced output spent (removed from the UTXO), and the latter being added to the UTXO.

Definition 2 [7]: A *block* consists of a transactions' list, a reference to the previous block and a nonce. Each block contains those transactions that the block creator (called the miner) has accepted in its memory-pool since the previous block.

A. BLOCKCHAIN TECHNOLOGY

Blockchain technologies are specifically designed to avoid single point of failures, i.e. those scenarios in which a single fault (either malicious or not) can affect the entire system by disrupting the provided service. These technologies solve this problem by replicating the server nodes and orchestrating their interaction with clients thus, achieving fault-tolerant services. As such, the fundamental property achieved by blockchain technologies is the state machine replication (SMR), which is defined as follows (we will use the Bitcoin's terminology for the sake of simplicity):

Definition 3 [7]: A set of miners achieves *state replication*, if all the miners execute a (potentially infinite) sequence of transactions $t_1, t_2, t_3, \ldots$, in the same order.

State replication is crucial to enforce the exact same state for all miners over time, while a set of transactions (issued by several users/wallets) is received and executed. Note that, miners are usually located on different machines to ensure that their eventual failures are independent. Although different in several aspects such as performances, permissions, provable security and computational completeness, any blockchain implementation satisfies the above definition. As an example, a central blockchain's tool that differs among implementations is the *consensus* algorithm [8]. It solves the following problem, which is crucial in designing an efficient SMR protocol. In the consensus problem, we consider a finite set of processes (or nodes in the network) $p_1, p_2, \ldots, p_n$ which communicate by exchanging messages. These processes could fail and we will consider the worst case: the byzantine failure. Initially, each process p_i is in an *undecided*

[1] https://coinmarketcap.com/

state and proposes a value v_i by broadcasting it to every other node. At the end, each node p_i will decide the value of its *decision variable* d_i. We can now formally define consensus as follows [9]:

Definition 4: A set of n processes $p_1, p_2, \ldots, p_n$ achieves *consensus* if the following properties hold:

- Agreement: the decision values of all the correct processes are the same;
- Integrity[2]: if the correct processes all proposed the same value v, then any correct process has set its decision variable to v;
- Termination: eventually each correct process sets its decision variable.

In the remaining part of this section we give a brief review of how standard ADS systems work and provide an overview on how we can build an ADS on top of the meta-data leveraged by a blockchain running a proof-of-work like [10] consensus protocol that generates local meta-data discarded at the time of block creation. We refer the reader to [7] for a formal and more complete treatment of blockchain's protocols.

B. ANOMALY DETECTION SYSTEMS

By recognizing and then discarding, sanitizing, or otherwise nullifying outliers input that might exploit security vulnerabilities, ADS often play a central role in many computer security systems [11]. Formally, an ADS can be defined as a couple (M, D), where M is the reference model describing the expected behavior while D is a similarity measure which specifies the actual behavior's deviation from M. Over the years, several ADS approaches have been proposed.

In statistical methods for anomaly detection, the system observes subjects' activities and generates different profile baselines to represent their behavior. Haystack was one of the earliest examples of statistical based ADS [12] which used a range of values that were considered normal and used to detect intrusions. Machine learning based prediction tools can be used to guess the next expected values; thus, they can be used in ADS to build the reference model by predicting normal incoming events, given the current ones. It is then possible to detect anomalies by selecting those next events which are not the ones anticipated by the prediction tools [13]–[15]. Machine learning approaches study algorithms that allow systems to derive general behaviors from data, and which can be either supervised or unsupervised. The first model is created from known clean data while the second is constantly analyzing data and modifying the behavior model without owning a previous one. For example, Spectrogram [16] is a machine learning based statistical ADS for defense against web-layer code-injection attacks orchestrated by a network situated sensor that dynamically assembles packets to reconstruct content flows, and learn to recognize legitimate web-layer script inputs. Taint-based techniques have been analyzed in ADS to avoid the false positives common issue. However, their applicability is limited by the need for accurate policies on the use of tainted data. Cavallaro *et al.* [17] developed a solution capable of detecting attack types that have been problematic for taint-based techniques, while significantly cutting down the false positive rate.

A preliminary report on the work in progress on BAD was published in [18]. In those two pages we just exposed the general idea. In this contribution, we experimentally prove its viability and formally define the related framework. Note also that BAD served as a baseline for filing a Nokia Bell Labs patent [19]—a clear sign of its innovative and viable approach, poised to have a concrete impact on both industry and research.

C. ADS CHALLENGES

ADS usually need to protect the reference model used to detect known and unknown threats [20], [21]. In host-based ADS (H-ADS) this database is stored locally while in network-based ADS (N-ADS) it can be either centralized on a trusted third party or distributed among the peers.

The problem of having centralized data-storage and management systems which are susceptible to breaches becomes even worse in truly distributed networks such as the ones leveraging blockchain technologies [22]. Furthermore, although a blockchain technology prevents several types of unexpected behaviors from malicious or compromised peers on a global scale, it does not eliminate attacks on a local scale. Indeed, local malfunctioning of the blockchain (see Section III) are discarded and cannot be used by others to recognize attack sequences that get reused over time by an attacker. As a result, ADS tools aimed at protecting blockchain-based systems cannot solely rely on information appearing within the mainstream chain but also need to take into account local contexts, and share such information on a global scale.

Table 1 lists some of the latest approaches in designing ADS systems on top of blockchain technologies [23]. We have grouped these approaches in Table 1 by highlighting (on the columns) four key properties as follow:

- Approach: describes whether the solution uses blockchain technologies to: i) build a *framework* for detecting anomalies; ii) as a simple (yet reliable) *storage system* to keep track of ADS data built by other tools; and, iii) as *other ADS approaches* that applied various techniques on top of blockchain meta-data;
- Attack: identifies the vector through which malicious data is introduced within the system, i.e. how the adversary tries to subvert the system or control sensitive data. It can either be on-chain or off-chain. The former identifies attacks using the blockchain data structure to inject malicious code, while the latter identifies those attacks which are carried outside the blockchain;
- Data usage (for short DU): this is a boolean flag that identifies those solutions that leverage blockchain meta-data, usually discarded by the p2p network, to better understand and identify anomalies within the system;

[2] In the literature, integrity is also called "validity".

TABLE 1. Related works on blockchain-based anomaly detection systems.

Solution	Approach	Attack	DU	DC
S. Iyer [24]	storage	off-chain	·	·
S. Sayadi [25]	storage	on-chain	·	·
Y. Mirsky [26]	storage	off-chain	·	·
M. Li [27]	storage	off-chain	·	·
O. Alkadi [28]	storage	off-chain	·	·
S. Morishima [29]	other	off-chain	·	·
Z. Il-Agure [30]	other	off-chain	✓	·
M. Salimitari [31]	framework	off-chain	·	·
X. Wang [32]	framework	on-chain	✓	·
B. Podgorelec [33]	framework	on-chain	✓	·
BAD (Our solution)	framework	on-chain	✓	✓

- Data creation (for short DC): unlike the above property that leverages blockchain data to analyze anomalies, this boolean property identifies those solutions enriching the standard blockchain meta-data with additional information that could help other nodes in identifying anomalies.

As shown in Table 1, although there are other works focusing on the study of ADS applied to the blockchain technology, to the best of our knowledge, BAD is the first approach that designes an ADS framework which not only works with the blockchain meta-data (forks being created and discarded over time) but also enriches it by sharing on a global scale all those information that are typically generated and stored on a local scale.

III. THREAT MODEL

The solution proposed in this paper has been designed to be resilient against any class of mass attacks where a malicious entity (usually in the form of a mass-targeting threat such as Botnets, collective hacking, etc) can append its own transactions within the blockchain to inject malicious code on multiple devices. However, for the sake of simplicity and clarity, we will use the well-known *eclipse attack* [5], [6] to provide an example of these attacks, and how our solution counters them.

Definition 5: A *fake transaction* is a blockchain transaction used as a side channel to deliver an unexpected message.

Definition 6: A *malicious transaction* is a special type of fake transaction in which the hidden message has the main purpose of attacking one or more peers within the network.

Definition 7: A *fake block* is a blockchain block that contains one or more fake/malicious transactions. Fake blocks can be either eventually discarded or accepted as part of the mainstream chain.

The standard blockchain network (used in Bitcoin) has been designed to be decentralized and independent of any public key infrastructure. Indeed, each node connects to 8 other nodes stored in a list that is obtained by querying DNS seeders. In an eclipse attack, the attacker infects a node's list of IP addresses, thus forcing the victim's node to connect to IP addresses controlled by the attacker. Furthermore, the attacker also aims at filtering and manipulating victim's incoming connections.

One way to execute an eclipse attack, is to repeatedly and rapidly forming unsolicited incoming connections to the victim by attacker's controlled IP addresses and then to wait until the victim restarts [34]. Hence, one challenge for the attacker is to control enough number of IP addresses in order to increase the probability that all the victim's outgoing connections will be directed to IP addresses controlled by him (see Section V). Once the attacker has monopolized all the victim's connections, he can filter incoming blocks and send his own *fake blocks* containing either *malicious transactions* as it has been done in ZombieCoin [35] (see Fig. 1). For the above attack to succeed, we assume the following attacker's capabilities:

- **Network Control**: the attacker can manipulate victims' connections in order to control their inbound and outbound traffic, thus being able to isolate them. This is a standard requirement for the eclipse attack;
- **Blockchain Control**: the attacker is capable of creating fake blocks which are sent to the victim. Their content is forged ad hoc by the attacker and usually contains a malicious payload.

a: LIVENESS OF THE SYSTEM

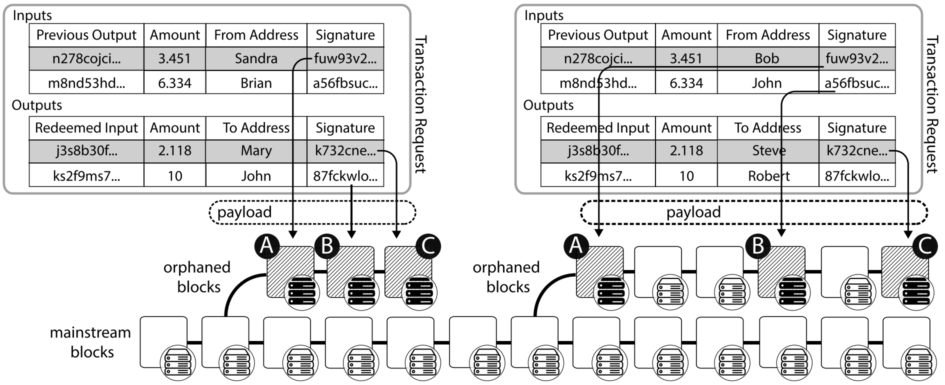
As described in Section V-A, we assume to have one or multiple powerful attackers who are able to perform eclipse attacks by targeting several victims. However, they have to complete in a finite time window. This means that we always assume that the victim(s) will eventually: i) recognize a fork, ii) synchronize with the mainstream blockchain technology and iii) share all the information collected during the eclipse attack with other peers in the network.

IV. BAD: A BLOCKCHAIN ANOMALY DETECTION SOLUTION

The core idea behind Blockchain Anomaly Detection (BAD) consists in providing a new decentralized system based on the blockchain technology which leverages all the information collected from past forks. In blockchain-based applications, forks become more important as the chances to create their evolution for malicious purposes get higher. The rationale behind this approach is that while attacks may happen only once within a single device, when they are repeated over time against other devices they usually keep behaving in the same way. Hence, by collecting information on previous attacks, it could be possible to black list them and to prevent them within those peers that have not been attacked yet. In the following we first report the rationale that inspired BAD and provide and example of its applicability, and later discuss the complete application stack of our solution.

A. BAD: RATIONALE AND EXAMPLE

In our solution, information regarding chain forks and their orphaned blocks, is discarded (as usually done in classical approaches). Indeed we collect, enrich and share such information with other peers in the network. Shared information



Inputs

Previous Output	Amount	From Address	Signature
n278cojci...	3.451	Sandra	fuw93v2...
m8nd53hd...	6.334	Brian	a56fbsuc...

Outputs

Redeemed Input	Amount	To Address	Signature
j3s8b30f...	2.118	Mary	k732cne...
ks2f9ms7...	10	John	87fckwlo...

Inputs

Previous Output	Amount	From Address	Signature
n278cojci...	3.451	Bob	fuw93v2...
m8nd53hd...	6.334	John	a56fbsuc...

Outputs

Redeemed Input	Amount	To Address	Signature
j3s8b30f...	2.118	Steve	k732cne...
ks2f9ms7...	10	Robert	87fckwlo...

FIGURE 1. Example of malicious code embedded within orphaned blocks.

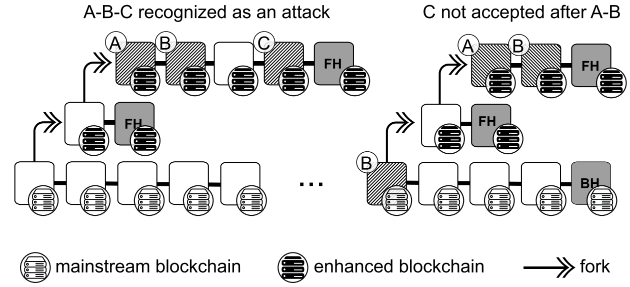


FIGURE 2. An overview of BAD being used as a tool to avoid known blockchain-based attacks to be repeated over time.

contains: i) the time at which the fork has started; ii) the time at which the fork has been detected; and, iii) the number and type of malicious transactions, if any, that has been identified within the fork. Fig. 2 shows a toy example of how we build our enhanced blockchain. The longer chain in the figure represents the mainstream chain (eventually agreed by all peers) with *block head* (BH) being the last blocks accepted. Shorter branches represent forks that happened in the past with *fork head* (FH) being the last blocks accepted before a new fork was created. Last, but not least, the figure also contains an example of malicious payload being spread through the blockchain. Such payload is composed by three transactions labeled $\{A, B, C\}$ which, as explained in Section III, can be either fake transactions or valid transaction embedding malicious code.

The collection of all fork-related information and the building of an enhanced blockchain made us able to design BAD as an ADS for blockchain-based applications. In fact, by having the enhanced blockchain agreed by all peers we only had to re-define (M, D) (see Section II-B) to model our ADS. Indeed in our solution, M is represented by the mainstream blockchain, thus describing the expected behavior, while $D(s)$ is represented by the fork(s), thus describing similarity measures and their deviation from M. It is then possible to learn that, as shown in Fig. 2, A-B-C have been previously labeled as an attack thus to prevent them from being re-executed on other peers.

Note that our solution is particularly efficient when the attacker, or the payload being spread, replicates the same operations (i.e. the transaction content) against every peer (for example, this is common in Botnet's attacks). In a general case, where attacks are crafted specifically for single targets, an additional ML/AI layer could be considered for comparing suspicious transactions with a set of malicious sequences (collected over time), in order to identify a potential attack and eventually prevent it (see Section VIII).

B. APPLICATION STACK

The standard blockchain application stack is structured in three layers: shared data, shared protocol and application.

Shared Data Layer: contains the core blockchain and its overlay network. It is still based on the core blockchain protocol but it is used to build networks (called sidechains [36]–[38]) that work in parallel to the mainstream chain to perform tasks that the mainstream chain cannot solve while still relaying on the same data structures. Whatever forms these overlay networks take, they all share the connection to the mainstream chain. Such a connection is used to bootstrap their own alternative solution by leveraging the mainstream peer-to-peer network;

Shared Protocol Layer: thanks to the blockchain it is now possible to develop decentralized applications with built-in data (transaction payload), validation processes, and transactions that are not controlled by any single entity;

Application Layer: applications built on top of the shared data layer and the shared protocol layer work very similarly to the ones we have nowadays. However, they inherit security, privacy and decentralization properties from the underlying blockchain technology. Hence, peers using these applications will be able to talk with each other and finally reach an agreement which is trusted even though no central authority has been used.

As shown in Fig. 3, BAD has been designed to be an ad hoc solution (i.e. a blockchain based application plug-in or a third party service) rather than being embedded within Bitcoin or any other specific blockchain application. The reason for such approach is that BAD does not rely on a specific blockchain

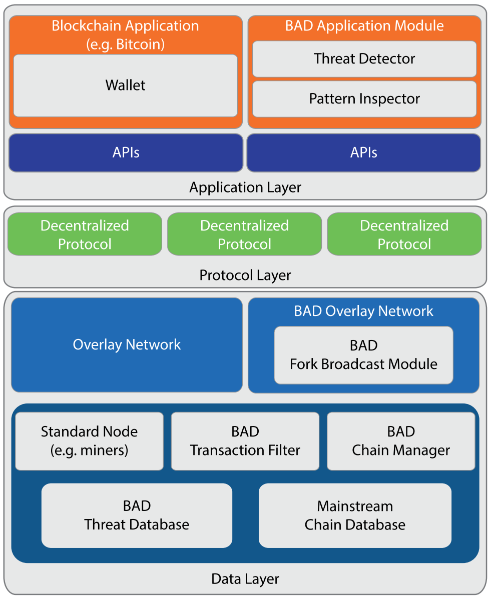


FIGURE 3. BAD application stack.

and can be instructed to detect attacks on any blockchain application. Indeed, the core Bitcoin elements such as the *wallet* and the *miner* do not contain BAD elements but just interact with them. Here, we describe each BAD's module and how it interacts with standard blockchain applications:

- Transaction Filter (Tx Filter): intercepts standard blockchain messages and forward them to both the *miner* and the *chain manager*, thus not interrupting the standard protocol. Furthermore, it allows the collection of transactions meta-data;
- Chain Manager: it is responsible to build our enhanced blockchain which, among the other elements, contains information on all forks that have been generated so far. It receives messages from the *transaction filter* and retrieves additional missing information from the *chain database* which finally stores our *enhanced blockchains*. Last but not least, the chain manager notifies the pattern inspector if the enhanced blockchain has been updated and some threat analysis has to be applied;
- Pattern Inspector: leverages the *chain database* to detect unexpected behaviors. The inspection on the forks can be done with any approach ranging from signatures to heuristic static analysis and it is aimed at finding sequences of transactions which were found to be dangerous in the past;
- Threat Detector: starting from the anomalies found by the *pattern inspector* this module performs root-cause analysis by exploiting past blockchain activities (past blocks and transactions within them) to roll back all the operations done by the victim. Afterwards, all the attack information are collected within a *threat database* which contains the information on all malicious patterns within the blockchain that have to be considered malicious (depending on the security policy being adopted).

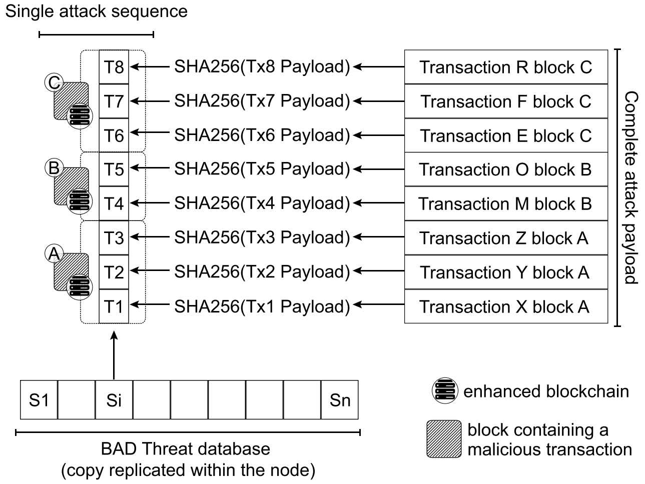


FIGURE 4. Implementation of the threat database in BAD.

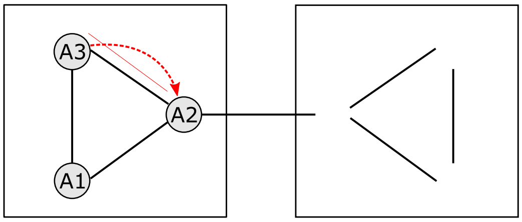


FIGURE 5. Eclipse Attack in Domain A.

Fig. 4 shows a simple implementation of BAD's threat database. Here, recalling the toy example given in Fig. 2 in which $\{A, B, C\}$ were found to represent chunks of a malicious payload, we show how this information is collected and later shared with other peers. BAD's threat database is basically a dynamic (i.e. not sized) array of array in which S_i represents the i-th attack sequence detected while T_i represents the hash of the i-th transaction which was found to contain part of the payload's attack sequence.

Information used to fill the threat database is provided by the pattern inspector and used by the transaction filter to avoid the repeating of known attacks. The filtering process is accomplished by the BAD's transaction filter module each time a new block is received and its overhead has been analyzed in Section VI-B.

V. EXPERIMENTAL TEST

In this section we show how BAD has been used in our experimental platform to prevent attacks across different networks thanks to the information collected from forks. The goal of this experiment was to detect forks on a given peer, that were caused by an eclipse attack, and then to share this information with other peers in order to build a reference model, aimed at detecting future occurrences of the same attack.

A. TESTBED

For simplicity, the testbed shown in Fig. 5 is only composed by two domains, A and B, that represent two separated private IP networks with a router between them. In domain A (*B respectively*), we have deployed two full nodes and one lightweight client as follow:

- Full Nodes: two active full nodes[3] *A1* (*B1* in domain B) and *A3* (*B3* in domain B) are deployed on a virtual machine with 4 GB of RAM with Linux Ubuntu 16.04 as a guest operating system. Both have been executed in regtest experimental mode,[4] i.e. a mode in which local testing environment can be created with instantaneous on-demand block generations and digital assets creation, without any real value. During the experimental tests, *A3* in domain A (*B3* in domain B) is assumed to be controlled by a malicious user;
- Client Node: as a lightweight client we used a Bitcoin Java BitcoinJ wallet (version 0.14.3)[5] running on a 4GB RAM PC with Windows 8.1 installed as a guest operating system. This wallet acts as the victim of the eclipse attack and is labeled as *A2* (*B2* in B).

A1 (*B1* in domain B) and *A3* (*B3* in domain B) are connected to each other, which means they can exchange blocks and agree on the longest chain—to do so we used on each node the following command: `bitcoin-cli -regtest addnode IPaddr add`. Nodes in domains A and B are initially synchronized on the same blockchain—this blockchain is generated using the command: `bitcoin-cli -regtest generate X`, that is meant to initialize *X* blocks in the blockchain.

B. ATTACK DETECTION AND PREVENTION

Based on the above testbed, we have implemented a real attack using bitcoin-cli commands. The attack aims at eclipsing a victim node and force it to accept some malicious blocks containing a payload. The attack, as well as the creation of our enhanced blockchain, has been implemented as follows:

1) eclipsing *A2* and forcing it to only communicate with *A3* which is controlled by the attacker;
2) stop *A3* from exchanging blocks with *A1* to avoid being detected by other nodes in the same domain. This has been implemented via executing: `bitcoin-cli -regtest addnode IPaddressofA1 remove` within *A3*;
3) make *A3* sending to *A2* three new blocks containing forged transactions. We have implemented this via the command: `bitcoin-cli -regtest generate 3`;
4) wait for *A2* to send the above fake blocks as connected to the previous blockchain header and representing the longest chain received so far. Assuming that the above three new blocks, created by *A3*, contains a malicious payload, we can conclude that *A2* is compromised at this step;
5) as the attack is completed, the eclipse on the victim is removed. Hence, *A2* starts again to communicate with other peers in the same domain, and to receive blocks from them which eventually forces *A2* to receive a longer chain that does not contain the above three fake blocks. At this point, and by leveraging on our BAD modules, *A2* is capable of keeping track of the malicious blocks received and to share this information broadcasting it to the other peers in all domains.

As a second phase we executed the same steps described above but this time within domain B. By leveraging BAD, and the information gathered so far from domain A, peers in domain B were able to detect and to prevent the attack from succeed. Indeed, we witnessed the (attempted) attack in domain B to behave as follows:

1) *B2* is eclipsed by forcing it to only connect to *B3*, here controlled by a malicious user;
2) *B3* generates three malicious blocks which contain, among the others, the same three malicious transactions used in the attack against *A2*;
3) unlike *A3*, *B3* has now the knowledge of some malicious blocks/transactions that resulted in another peer being compromised. Indeed, as also shown in Fig. 2, BAD is able to detect blocks that are different but contains the same transactions (or a subset), in the same order, as previously received by *A2*.

The final result is the prevention of the complete attack as only a small subset of the malicious transactions is accepted (in our example accepted by *B2*) before BAD recognizes them as malicious. As done by *A2*, also *B2* will share the information with other peers once it realizes that it was previously mining and elaborating on blocks that belonged to a malicious fork.

VI. OVERHEAD ANALYSIS

The core elements introduced by BAD on the classical Bitcoin protocol are the broadcast of brand new forks, their orphaned blocks, as well as the detection of malicious transactions on new received blocks. In this section, we analyze the introduced bandwidth overhead to show that our solution is scalable and thus deployable within the standard Bitcoin network. In particular, the results of our analysis show that our system has minimal bandwidth consumption in comparison with the one consumed by standard nodes.

A. BANDWIDTH OVERHEAD

We have analyzed the overhead introduced by our solution in the worst-case scenario, i.e. the whole global Bitcoin fork activity to affect one single node named *NX*. Our overhead is then defined as the amount of bandwidth that *NX* consumes due to the fork broadcast introduced in BAD. To this aim, and to be rooted on real data, we have considered the maximum number of orphaned blocks discarded by the Bitcoin community during last year. We are interested in the total number of orphaned blocks because it includes those used to

[3] https://bitcoin.org/en/full-node#what-is-a-full-node

[4] https://bitcoin.org/en/developer-examples#regtest-mode

[5] https://bitcoinj.github.io

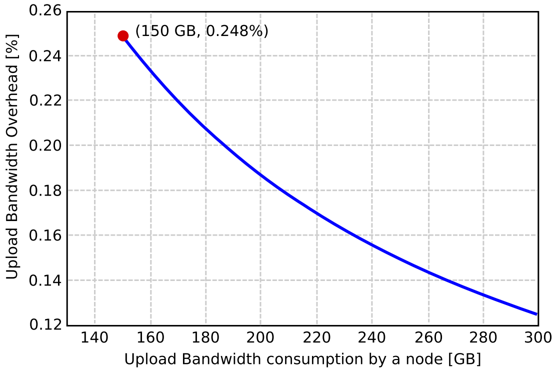


FIGURE 6. Overhead introduced by the system as a function of the bandwidth consumption of a node.

attack the victims (see Section III). Furthermore, we assume this number to have a small variance since a smart adversary, to stay hidden in the network, would not create an anomalous number of orphaned blocks. A more abstract, and less constrained, analysis is given in Section VI-B.

To analyze BAD's overhead, we have designed the p2p network surrounding our *NX* node. By construction, nodes in the Bitcoin network create a random graph with randomness emerging from the selection of outgoing connections. In the vanilla Bitcoin protocol, each node attempts to keep a minimum of 8 outgoing connections at all time. However, it has been observed that, on average, a Bitcoin node has 32 outgoing connections [39]. Furthermore, the total number of orphaned blocks discarded during 2016[6] was 141 with a maximum block size of 0.993201 MB. As such, in our worst-case scenario, we consider all those 141 orphaned blocks (of the maximum size) to be collected and re-distributed in broadcast by *NX*. To broadcast all these blocks with their transactions, *NX* would send broadcast messages to its neighbors, which sum up to the global size of $32 \times 0.993201 \times 141 = 4.481$ GB per year. It is important to highlight that the total number of orphaned blocks is independent of the node's bandwidth. Hence, our worst-case scenario can be applied to any node: from lightweight SVP clients to relay nodes or miners. Furthermore, the total node/month upload bandwidth could vary according to nodes capabilities and ISP resources: it could require an initial 150 GB/month of uploaded data (which is the minimum recommended upload data plan to run a Bitcoin core[7]) and reach values up to 300 GB/month or more.

Fig. 6 plots the result of our BAD's overhead (*Ovh*) analysis which is approximated by the following formula:

$$Ovh = \frac{\text{BAD data broadcast (per year)}}{\text{total data exchanged (per year)}} = \frac{4.481}{m \times 12} \qquad (1)$$

where m is the average bandwidth consumption of a node per month. Fig. 6 shows the maximum overhead introduced in the case of 150 GB of data upload consumption, resulting in a bandwidth overhead of only 0.248%. This latter figure supports the fact that BAD is a lightweight security add-on that can be smoothly deployable in the standard Bitcoin network.

B. COMPLEXITY

In the previous section we studied BAD's overhead in the worst case, i.e. with an attacker using Bitcoin's forks to spread malicious code. However, statistics and real data used for such analysis refer to natural forks appeared over time in the network due to its delay.

In thus section, we analyze a more general use case where the attacker creates as many blocks as needed (thus also generating more forks in the system). The result, as shown in the remaining of this section, is that BAD's bandwidth overhead, in the worst case, can only be proportional (up to a constant factor in real cases) to the size k of our Threat Database $\mathcal{T}$. Let $S_1, \ldots, S_k$ be the malicious transaction sequences of k attacks detected and stored in $\mathcal{T}$. Each malicious sequence S_i has a length of ℓ_i transactions injected by the attacker to complete attack i. We call *partial sequence* (PS_i, j) a subsequence of S_i starting from the first transaction and ending with the j-th transaction of S_i. Note that (PS_i, ℓ_i) represents the full attack i. For each attack i we can have at most $\ell_i - 1$ distinct partial subsequences. Each node in the network maintains a set U of partial transactions. Given that $H(t)$ is the hash of a transaction t, every time t is analyzed by a node, BAD performs two actions:

1) If there is a partial sequence $(PS_i, j) \in U$ such that $(PS_i, j)||H(t) = (PS_i, j+1)$, we replace (PS_i, j) with $(PS_i, j+1)$ in U. Here $||$ is the standard concatenation function.
2) If $H(t)$ represents the first block of a sequence S_i, then we insert $(PS_i, 1)$ into U.

Finally, BAD checks if there is a (PS_i, ℓ_i) in U and, in that case, discards the transaction t. While the correctness of this approach follows from the construction, the additional computational cost (per transaction) incurred by each node in the network can be derived. Note that, in the worst case (which is when every transaction of every attack has the exact same hash), every transaction will create a new partial sequence $(PS_i, 1)$, $\forall i$, plus it will increase at most $\ell_i - 1$ existing partial sequences in U for each attack i. This translates in the following number of steps:

$$W(t) = k + \sum_{i=1}^{k}(\ell_i - 1) = \sum_{i=1}^{k} \ell_i$$

Since (in a real scenario) each attack sequence is no longer than a constant c of transactions, the total work $W(t)$ for a given transaction will be at most $c \cdot k = O(k)$ where $k = |\mathcal{T}|$. In case the size of $\mathcal{T}$ grows very quickly, pruning techniques can be adopted to adjust its dimension. For example, old or infrequent attacks could be discarded in favor of newly discovered ones.

[6] https://blockchain.info/charts/n-orphaned-blocks

[7] https://bitcoin.org/en/bitcoin-core/features/requirements

VII. DISCUSSION

The solution proposed in this work requires an attacker capable of *pushing* fake blocks into his/her victim, i.e. to make the latter believing that some fake blocks received have been already accepted within the mainstream chain. In blockchain-based applications, this outcome can be achieved with a broad range of attacks spanning from owning 51% of the whole peer-to-peer network, to leveraging the structure of the overlay network to eclipse the victim. Blockchain-based applications make large use of overlay networks [40], i.e. connections forming a graph upon which a distributed application is implemented, as they allow to deploy network functionalities without changing the underlying infrastructure.

As described in Section V, the experimental tests provided to support our solution have been obtained by implementing an eclipse attack on our blockchain network. This required some bitcoin-cli commands that forced our victim node into adopting the malicious nodes as its peers, thus accepting all blocks received by them. The eclipse attack deployed on our network was quite easy to accomplish due to the limited size of our network. However, the state of art on distributed systems shows that a wide range of countermeasures and defense techniques can be adopted against such attacks. The solutions proposed by Castro *et al.* [41] based on constrained routing tables as well as the one proposed by Simgh *et al.* [42] based on neighbor anonymous auditing are just some example describing how the eclipse attack can be prevented. Although this may suggest that the solution proposed in this work is limited since not easily deployable in real networks, it should be highlighted that the above defense techniques against eclipse attacks, are used to either make some strong assumption on the network size/structure or to prevent optimizations like *proximity neighbor selection* [43]: an important and widely used technique to improve overlay efficiency. Last but not least, the continuous development of new peer-to-peer protocols, mining algorithms and consensus schemes can make new blockchain applications more exposed to eclipse attacks. In this latter case, BAD would be easily adoptable to counter such a threat, as well as to provide a customizable platform to counter further threats.

VIII. CONCLUSION

In this paper, we proposed BAD: the first Blockchain Anomaly Detection solution. In particular, BAD allows to detect anomalous transactions and to prevent them from being further spread. Indeed, while forks can naturally appear in the blockchain life cycle due to the network delay, they can also be artificially forged by attackers and used to spread malicious activities within the chain. BAD enables the prevention of repeated attack occurrences by collecting malicious activities and building a threat database which is distributed (thus avoiding any central point of failure), tamper-proof, trusted (any behavioral data is collected and verified by the majority of the network), and private.

We detailed BAD, and provided an analysis of its overhead, as well as a prototype implementation, demonstrating its effectiveness in detecting, for instance, the dreadful eclipse attack. The achieved results show the quality and viability of our solution, that could also be a starting point for further investigation in this domain.

As for future work, we envisage the adoption of efficient ML techniques to further refine the capability to detect attacks, should these latter ones show polymorphic features in order to escape detection.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their suggestions, that helped to improve the quality of the manuscript. The publication of this article was funded by the Qatar National Library (QNL), Doha, Qatar and award NPRP11S-0109-180242 from the Qatar National Research Fund (QNRF), member of Qatar Foundation. The information and views set out in this publication are those of the authors and do not necessarily reflect the official opinion of QNL and QNRF.

REFERENCES

[1] S. Greenstein, ''The aftermath of the dyn DDOS attack,'' *IEEE Micro*, vol. 39, no. 4, pp. 66–68, Jul. 2019.

[2] R. Di Pietro and L. V. Mancini, *Intrusion Detection Systems*, 1st ed. New York, NY, USA: Springer, 2008, doi: 10.1007/978-0-387-77265-3.

[3] H. S. Javitz and A. Valdes, ''The SRI IDES statistical anomaly detector,'' in *Proc. IEEE Comput. Soc. Symp. Res. Secur. Privacy*, May 1991, pp. 316–326.

[4] A. Zohar, ''Bitcoin,'' *Commun. ACM*, vol. 58, no. 9, pp. 104–113, Aug. 2015.

[5] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach, ''Eclipse attacks on overlay networks: Threats and defenses,'' in *Proc. 25th IEEE Int. Conf. Comput. Commun. (INFOCOM)*,Apr. 2006, pp. 1–12.

[6] K. Nayak, S. Kumar, A. Miller, and E. Shi, ''Stubborn mining: Generalizing selfish mining and combining with an eclipse attack,'' in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 305–320.

[7] R. Wattenhofer, *The Science of the Blockchain*, 1st ed. North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2016.

[8] M. J. Fischer, ''The consensus problem in unreliable distributed systems (a brief survey),'' in *Foundations of Computation Theory*, M. Karpinski, Ed. Berlin, Germany: Springer, 1983, pp. 127–140.

[9] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed. Reading, MA, USA: Addison-Wesley, 2011.

[10] I. M. Ali, M. Caprolu, and R. D. Pietro, ''Foundations, properties, and security applications of puzzles: A survey,'' *ACM Comput. Surv.*, vol. 53, no. 4, pp. 1–38, Aug. 2020, doi: 10.1145/3396374.

[11] V. Chandola, A. Banerjee, and V. Kumar, ''Anomaly detection: A survey,'' *ACM Comput. Surv.*, vol. 41, no. 3, p. 15, Jul. 2009.

[12] S. E. Smaha, ''Haystack: An intrusion detection system,'' in *Proc. 4th Aerosp. Comput. Secur. Appl.*, Sep. 1988, pp. 37–44.

[13] R. Halavati, S. B. Shouraki, P. Esfandiar, and S. Lotfi, ''Rule based classifier generation using symbiotic evolutionary algorithm,'' in *Proc. 19th IEEE Int. Conf. Tools Artif. Intelligence(ICTAI)*, vol. 1, Oct. 2007, pp. 458–464.

[14] M. E. Cintra, E. R. Hruschka, H. de A. Camargo, and M. do Carmo Nicoletti, ''Fuzzy rule base generation through genetic algorithms and Bayesian classifiers a comparative approach,'' in *Proc. 7th Int. Conf. Intell. Syst. Design Appl. (ISDA)*, Oct. 2007, pp. 315–322.

[15] E. Hinojosa Cardenas and H. A. Camargo, ''Multiobjective genetic generation of fuzzy classifiers using the iterative rule learning,'' in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Jun. 2012, pp. 1–8.

[16] Y. Song, A. D. Keromytis, and S. Stolfo, ''Spectrogram: A mixture-of-Markov-chains model for anomaly detection in Web traffic,'' in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2009, pp. 1–5.

[17] L. Cavallaro and R. Sekar, ''Taint-enhanced anomaly detection,'' in *Information Systems Security*, S. Jajodia and C. Mazumdar, Eds. Berlin, Germany: Springer, 2011, pp. 160–174.

[18] M. Signorini, M. Pontecorvi, W. Kanoun, and R. Di Pietro, "ADvISE: Anomaly detection tool for blockchain systems," in *Proc. IEEE World Congr. Services (SERVICES)*, Jul. 2018, pp. 65–66.

[19] M. Signorini, R. Di Pietro, and W. Kanoun, "Blockchain-based security threat detection method and system," U.S. Patent App. 019 182 284 A1, Jun. 13, 2019.

[20] S. S. Murtaza, A. Hamou-Lhadj, W. Khreich, and M. Couture, "Total ADS: Automated software anomaly detection system," in *Proc. IEEE 14th Int. Work. Conf. Source Code Anal. Manipulation*, Sep. 2014, pp. 83–88.

[21] J. Kim and H. S. Kim, "PBAD: Perception-based anomaly detection system for cloud datacenters," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, Jun. 2015, pp. 678–685.

[22] J. J. Xu, "Are blockchains immune to all malicious attacks?" *Financial Innov.*, vol. 2, no. 1, Dec. 2016.

[23] N. A. Dawit, S. S. Mathew, and K. Hayawi, "Suitability of blockchain for collaborative intrusion detection systems," in *Proc. 12th Annu. Undergraduate Res. Conf. Appl. Comput. (URC)*, Apr. 2020, pp. 1–6.

[24] S. Iyer, S. Thakur, M. Dixit, R. Katkam, A. Agrawal, and F. Kazi, "Blockchain and anomaly detection based monitoring system for enforcing wastewater reuse," in *Proc. 10th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2019, pp. 1–7.

[25] S. Sayadi, S. Ben Rejeb, and Z. Choukair, "Anomaly detection model over blockchain electronic transactions," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 895–900.

[26] Y. Mirsky, T. Golomb, and Y. Elovici, "Lightweight collaborative anomaly detection for the IoT using blockchain," *J. Parallel Distrib. Comput.*, vol. 145, pp. 75–97, Nov. 2020.

[27] M. Li, K. Zhang, J. Liu, H. Gong, and Z. Zhang, "Blockchain-based anomaly detection of electricity consumption in smart grids," *Pattern Recognit. Lett.*, vol. 138, pp. 476–482, Oct. 2020.

[28] O. Alkadi, N. Moustafa, and B. Turnbull, "A review of intrusion detection and blockchain applications in the cloud: Approaches, challenges and solutions," *IEEE Access*, vol. 8, pp. 104893–104917, 2020.

[29] S. Morishima, "Scalable anomaly detection method for blockchain transactions using GPU," in *Proc. 20th Int. Conf. Parallel Distrib. Comput., Appl. Technol. (PDCAT)*, Dec. 2019, pp. 160–165.

[30] Z. Il-Agure, B. Attallah, and Y.-K. Chang, "The semantics of anomalies in IoT integrated BlockChain network," in *Proc. 6th HCT Inf. Technol. Trends (ITT)*, Nov. 2019, pp. 144–146.

[31] M. Salimitari, M. Joneidi, and M. Chatterjee, "AI-enabled blockchain: An outlier-aware consensus protocol for blockchain-based IoT networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[32] X. Wang, J. He, Z. Xie, G. Zhao, and S.-C. Cheung, "ContractGuard: Defend ethereum smart contracts with embedded intrusion detection," *IEEE Trans. Services Comput.*, vol. 13, no. 2, pp. 314–328, Apr. 2020.

[33] B. Podgorelec, M. Turkanović, and S. Karakatič, "A machine learning-based method for automated blockchain transaction signing including personalized anomaly detection," *Sensors*, vol. 20, no. 1, p. 147, Dec. 2019.

[34] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *Proc. 24th USENIX Conf. Secur. Symp.*, Berkeley, CA, USA, 2015, pp. 129–144.

[35] S. T. Ali, P. McCorry, P. H.-J. Lee, and F. Hao, "Zombiecoin: Powering next-generation botnets with bitcoin," in *Financial Cryptography and Data Security*, M. Brenner, N. Christin, B. Johnson, and K. Rohloff, Eds. Berlin, Germany: Springer, 2015, pp. 34–48.

[36] R. M. Parizi, S. Homayoun, A. Yazdinejad, A. Dehghantanha, and K.-K.-R. Choo, "Integrating privacy enhancing techniques into blockchains using sidechains," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2019, pp. 1–4.

[37] B. N. Musungate, B. Candan, U. C. Cabuk, and G. Dalkilic, "Sidechains: Highlights and challenges," in *Proc. Innov. Intell. Syst. Appl. Conf. (ASYU)*, Oct. 2019, pp. 1–5.

[38] P. Gazi, A. Kiayias, and D. Zindros, "Proof-of-Stake sidechains," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 139–156.

[39] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Proc. IEEE P2P*, Sep. 2013, pp. 1–10.

[40] E. Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Commun. Surveys Tuts.*, vol. 7, no. 2, pp. 72–93, 2nd Quart., 2005.

[41] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," *ACM SIGOPS Operating Syst. Rev.*, vol. 36, no. SI, pp. 299–314, Dec. 2002.

[42] A. Singh, M. Castro, P. Druschel, and A. Rowstron, "Defending against eclipse attacks on overlay networks," in *Proc. 11th Workshop ACM SIGOPS Eur. Workshop: Beyond PC (EW11)*, 2004, p. 21.

[43] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The impact of DHT routing geometry on resilience and proximity," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun. (SIGCOMM)*, 2003, pp. 381–394.

MATTEO SIGNORINI is currently a Cyber Security Research Engineer with Nokia Bell Labs, France, where he moved in 2016, after a computer scientist academic background, concluded with a Ph.D. thesis, ranked top class at the Security CyberCamp 2014. He has coauthored over 20 works, including conference and journal papers, patents, and book chapters. His main research interests include privacy and security for distributed systems, with a special focus on State Machine Replication, including Blockchain technologies. He won the Cisco Security Grand Challenge in 2014 and was awarded with a Nokia Bell Labs distinguished research activity and two Nokia Bell Labs top inventor awards over the latest three years.

MATTEO PONTECORVI received the Ph.D. degree in computer science from the University of Texas at Austin. He joined the Cybersecurity Group, Nokia Bell Labs (Paris-Saclay), in 2017. Since then, he has been working on Blockchain technologies and distributed computing. He is also interested in algorithms, cryptography, and graph theory.

WAËL KANOUN held the position of the Head of R&D Cyber Security, while working at Nokia Bell Labs. He is currently the Head of Cybersecurity Solutions for Thales in the Middle East, and the International Lead of the Cybersecurity for Ground Transportation vertical at Thales. His domain expertise covers cybersecurity programs and frameworks, risk management, architectures, threat and vulnerability assessment, cyber detection and monitoring, authentication, applied cryptography, automated and dynamic cybersecurity assessment, R&D programs, and intellectual property. His experience spans various OT and IT systems for critical infrastructures such as telecom, ground transportation, defense, utilities (electricity and water), as well as public safety. His work is extensively published (over 20) in peer-reviewed international conferences and journals, and led to over 10 patent applications. He has served on review committees of several international conferences and journals, and has given cybersecurity lectures in French Universities and Graduate Schools.

ROBERTO DI PIETRO (Senior Member, IEEE) was Global Head for Security Research at Nokia Bell Labs, and also an Associate Professor (with tenure) of Computer Science with the University of Padova, Italy. He is currently a Full Professor in Cybersecurity at HBKU-CSE. He has been working in the security and privacy field for over 23 years, leading both technology-oriented and research-focused teams in the private sector, government, and academia (MoD, United Nations HQ, EUROJUST, IAEA, and WIPO). His main research interests include security and privacy for wired and wireless distributed systems, i.e., (Blockchain technology, Cloud, IoT, OSNs); virtualization security, applied cryptography, computer forensics, and data science. Other than being involved in M&A of start-up—and having founded one (exited)—he has been producing over 220 scientific papers and over 20 patent applications over the cited topics. He has coauthored three books, edited one, and contributed to a few others. He is an ACM Distinguished Member and he is serving as an AE for ComCom, ComNet, PerCom, *Journal of Computer Security*, and other International journals. He was awarded the Chair of Excellence (2011–2012) from University Carlos III, Madrid. In 2020, he received the Jean-Claude Laprie Award for having significantly influenced the theory and practice of Dependable Computing.