International Journal of Pattern Recognition and Artificial Intelligence

A Distributed Solution to Detect Anomalous Smart Contract within Blockchain Networks

William S. Ventura*

Whiting School of Engineering
Johns Hopkins University, Baltimore, MD ,USA †
wventur1@jh.edu‡

Keywords: Blockchain; Smart Contracts; Artificial Intelligence; Anomaly Detection.

1. Introduction

In the past few decades, data has continuously established itself as an integral component of everyday life. With the *Internet of Things* (IOT) revolution, an influx of smart devices capable of communicating and interacting with one another via Internet entered the global market.[15] This digital revolution facilitated the growth of data-driven applications; providing solutions to wireless communications, file-sharing, finance, and consumer goods.[18] The emergence of these smart data-driven applications placed an emphasize on user experience; promoting task efficiency aimed to make life easier for the user while generating a momentous amount of data. However storage of this ever-expanding amount of data poses serious problems, as well as security concerns based on its communication methods. Blockchain technology (BT), which has a distributed database network, is an optimistic solution to these issues.

Satoshi Nakamoto coined the term BT in 2008; a decentralized architecture that contained time-stamped series of immutable records managed by a cluster of distributed computers. Originally developed as a support tool for Bitcoin due to BT's three main characteristics (*immutability, decentralization, transparency*), it allowed untrusted peers in open (i.e permission-less) communities transparent access of shared databases without the need to access trusted third parties.[15,25] While BT ensured security and privacy issues seen in prior application architectures, some vulnerabilities have emerged post its implementation. Smart Contracts (SC), which are computer programs stored on the blockchain that automatically execute agreement terms between parties when conditions are met, are a vital component

*East 57th Street New York, NY, 10022
†Maryland, Whiting School of Engineering, 3400 N Charles Street, MD 21218, USA
‡wventur1@jh.edu

of BT. To illustrate, *Reentrancy* attacks occur between two smart contracts, where the code in a vulnerable contract is exploited by the attacking smart contracted to drain it of its funds. Meanwhile other attacks have begun to become increasingly sophisticated such as *majority* attacks (51% percent attacks), that give controlling parties power to alter the blockchain, or *Sybil* attacks for fake identity generation that aim to control the consensus by gaining majority influence in the network.[1 14]

To handle these aforementioned issues, designing efficient and effective algorithms to analyze massive amounts of data from blockchain-based smart applications is in dire need. Machine learning (ML) is highly prevalent today and could be used to detect intrusions and attack patterns within these blockchain-based smart applications. In the communication network of blockchain-based smart applications, some of the the security issues are handled at the network layer (e.g. malicious SC/packets) while others are handled at the application layer (e.g. malware).[12] At the network layer, malicious SC can be used to execute attacks such as *Reentrancy* or *Sybil*. To avert these issues, header data from SC can be analyzed through real-time application of ML models using historical data. In any smart application it is crucial that data should be secure. BT ensures data security but to build confidence, usage of ML models to predict unsecured nodes based on previous data is essential.

Research contributions

Though several research works exist to address security issues posed by malicious SC, to the best of current knowledge none have achieved scalability nor have been exploited to its full potential. This paper aims to take into consideration previous works for detecting malicious SC and propose a enhanced solution that is able to resolve scalability issues to provide a comprehensive distributed ADS to a blockchain network.In doing so, methodology from previous related works will be taken into consideration while assessing solutions to scalability of such a system on the blockchain network. Such approaches to resolving scalability issues currently include a cloud computing infrastructure and refinement in protocol layers.

Organization

This paper is organized as follows: Section 2 provides a background on blockchain technology as well as related previous works on anomaly detection systems. Section 3 describes the methodology for initial data collection. Section 4 & 5 introduces the proposed solution and experimental results respectively. Section 6 addresses potential issues and limitations of the system. Lastly, Section 7 concludes the paper and introduces future work.

2. Technology background and related work

2.1. *Blockchain*

Blockchain technology consists of blockchains, a set of immutable records that are cryptographically linked together for the purpose of auditing and acts similarly to an accounting ledger.[18,24] Records previously recorded on the blockchain ledger are *immutable*, and new records are required to be verify by a trusted party. There is no centralized party to verify these new blocks (set of records), instead it is checked by a decentralized structure of nodes that contain a copy of the ledger. It is by linking validated blocks that forms the blockchain, with the current block containing the hash of the previous block and so on, as shown in Figure 3.[10]

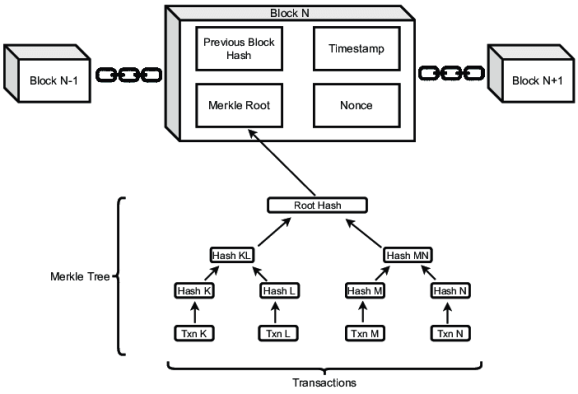

Fig. 1. Structure of an Ethereum blockchain node.

Older blocks cannot be modified, if it were to be changed in anyway; the hash associated with it would have to change and distributed to all subsequent blocks. This structure allows the blockchain to be traceable and ensures immutability. A copy of the ledger is available to every party in the network; therefore users can cross verify any changes that occur. With every new addition of a block, the copy of the blockchain is updated for all the users.

2.1.1. *Cryptographic hash*

BT technology uses a cryptographic secure hash algorithm (SHA) such as SHA-256 and SHA-516 to preserve data integrity within the block, for which each block is given a unique hash value. Bitcoin utilizes double SHA-256, while Ethereum uses Keccak-256 and Keccak-512. SHA is a collision-resistant algorithm that prevents production of the same output (hash value) given two different inputs and can be utilized to ensure data consistency.[13] Initially developed as a component of the

Digital Signature Standard (DSS) to produce signatures, consensus algorithms can be used to validate blocks. The type of blockchain, such as public, private, and consortium blockchain determine the algorithm selected.[18] To ensure consensus among the nodes, the selected algorithm must be able to efficiently use resources and tolerate a degree of safety in the event of an attack.[18,20]

2.1.2. *Smart contracts*

Smart contracts are programs (set of codes) that are executed on the blockchain and add blocks whenever certain conditions are met. It translates legal contracts into programs to enforce the legal contract between parties onto the blockchain.[19] The primary design of SC is to provide high-caliber security while reducing costs and delays associated with traditional contracts.[18] They act similarly to stored-procedures in relational databases, governing transactions that are either fully or partially executed.

2.1.3. *Protocol*

This paper only considers permission-less blockchain technologies(e.g. Bitcoin, Ethereum, and Solana), where there is a race between peers to mine blocks and raise potential forks.[a] Concepts and definitions from 22 are described below, followed by a description of the blockchain protocol.

An *input* is defined as a tuple consisting a reference to a previously created output and arguments for a spending condition, allowing the transaction creator to spend the referenced output. *UTXO* is defined as the set of unspent transaction outputs.

An *output* is defined as a tuple consisting of an amount of cryptocurrency (e.g. Ethereum), and a spending condition. Spending condition is typically a valid signature associated with a private key of the spender's address, it could also be a script that can be exploited by an attacker.

Definition 1. [22] *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. [22] *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.*

One of the primary purposes of BT is to avoid single point of failures, such as scenarios where a single fault can disrupt the provided service by affecting the en-

[a]This paper adopts the same notion used in Ref.22 to describe blockchain and state machine replication protocols.

tire system. Replication of server nodes and synchronization of interactions with clients resolve these issues, achieving fault-tolerant services. State machine replication (SMR) is a foundational characteristic achieved by such blockchain technologies, it is defined as the following:

Definition 3. [22] *A set of miners achieves state replication, if all the miners execute a (potentially infinite) sequence of transactions t_1, t_2, t_3, . . ., in the same order.*

State replication is crucial to invoke that exact same state to all miners across time, while a block of transactions (generated by various parties/wallets) are received and processed. These miners are distributed across different machines to ensure independence between eventual failures and the current blockchain. While blockchains differ in several aspects, any implementation of a blockchain satisfies the definition above. One blockchain tool that differs across different implementations is the *consensus* algorithm.[5] The *consensus* algorithm solves the following problem, which is crucial in designing efficient SMR protocols. The problem is defined as such: consider a finite set of processes (or nodes in a network), p_1, p_2, . . ., p_n that communicate by exchanging messages and could fail such that the worst case is the byzantine failure. Initially, each process p_i is in an *undecided* state and proposes a value v_i by broadcasting it to every associated node. At the end, each node p_i will decide the value of its *decision variable* d_i. Consensus is formally defined as such[3]:

Definition 4. *A set of n processes p_1,p_2,...,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.*

The remainder of this section gives a brief overview on how standard Anomaly Detection Systems (ADS) function and provide an overview of related previous works. [b]

2.2. *Anomaly detection systems*

To address aforementioned threats presented by the *IoT* revolution, *Intrusion Detection Systems* (IDS) were been developed in the past as tools aimed towards reinforcing the security of complex systems and networks. This is performed via capturing, monitoring, and analyzing users' data traffic, or in other terms the behaviors associated with the user. [4] This type of approach aims to build attack models and mitigation strategies, typically based on log analysis and data correlation. Current IDS are classified into two classes based on their approach: signature recognition

[b]For a comprehensive description of blockchain protocols, please refer to Ref. 22.

and anomaly behavior. [8] Signature recognition leverages databases where signatures of well-known attacks are matched and used to build a reference model to detect future attacks, as such there are caveats for performance in the presence of new attacks with unknown signatures. Alternatively, anomaly detection aims to build models of normal behaviors and raise alerts when there are deviations in behavior from its baseline. Hence, the goal of an *Anomaly Detection System* (ADS) is to construct normal behavior models and present it new behaviors to assess its deviation from the reference model.

ADS has an essential role in many computer security systems, by identifying and handling inputs that could potentially exploit security vulnerabilities. ADS is formally defined as such:

Definition 5. *A couple (M,D) where M is the reference model describing the behavior and D is the similarity measure specifying the actual behavior's deviation from M.*

One of the earliest approaches to ADS was Haystack's statistical based ADS that utilized a range of values considered normal to detect intrusions.[16] In recent years there has been an increasing usage of machine learning to generate prediction based models to predict the next expected values; hence, such application can be used in ADS to build the reference model given current events or behaviors. Anomalies could then be detected by identifying future events that were not anticipated by the model. Machine learning allows for systems to derive general models through three different approaches: supervised; unsupervised; and reinforcement learning. Supervised learning generate models from known clean data while unsupervised learning is constantly analyzing known data and finding hidden structures in it. Reinforcement learning on the other hand incorporates an intelligent agent to interact with its environment and through trial and error learn the optimal behavior in the environment that maximizes reward. One notable application of ML in ADS is Song et al. (2009) Spectrogram, an ML based statistical ADS for defense against web-layer code-injection attacks orchestrated by network situated sensors that reconstruct web traffic.[17] However, practical deployment of ADS has been limited due to false positive. To handle the false positive issue taint-checking techniques have been analyzed in ADS. Cavallaro et al. (2011) were able to significantly cut down the false positive rate by developing a solution capable of detecting an attack types that have been problematic for taint-based techniques.[2] Further advancements in ADS also include, Wu & Ortiz (2021) which sought application of reinforcement and active learning in ADS, outperforming unsupervised and semi-supervised approaches.

2.3. *Challenges*

In order to maintain the integrity of ADS, it is crucial to protect the reference model used to detect known and unknown attacks. [11] This database can either be stored locally as seen in host-based ADS (H-ADS), or in the case of network-

based ADS (N-ADS) it is distributed among peers or centralized on a trusted third party. [15] More importantly, centralized data-storage and management systems are susceptible to hacking, intrusions and breaches.[23] While BC's distributed consensus mechanism prevents hacking from malicious or compromised peers on a global scale, it does not eliminate attacks at a local scale. Oftentimes, BC malfunctioning at the local scale is discarded and users are unable to use such information to recognize subsequent identical attack sequences reused by the attacker. Therefore, any ADS implementation cannot solely rely on information from the main branch of the blockchain but must also consider local incidents and distribute such information on a global scale.

Table 1 lists some of the recent approaches to designing ADS and integrating it on top of blockchain technologies. These approaches have been grouped by four key properties defined as such:

- Approach: describes whether the proposed solution uses BT to: i) build a *framework* for detecting anomalies; or ii) as an *other* ADS approach that applies multiple techniques on top of BC meta-data;
- Attack: identifies how malicious data is introduced into the system. *On-chain* attacks are signified as attacks that use the blockchain data structure to implant malicious code. *Off-chain* on the other hand identifies attacks that are carried outside of the blockchain.
- Data usage: identifies solutions that makes use of blockchain meta-data that is typically discarded by the network to assess and detect anomalies within the system.
- Data creation: identifies solutions that enrich the standard blockchain meta-data by distributing additional information to the main branch so other nodes can identify anomalies.

Table 1. Related works on blockchain anomaly detection systems

Solution	Approach	Attack	Data Usage	Data Creation
Alkhalifah et al. [1]	other	on-chain	✓	·
Wang et al. [21]	framework	on-chain	✓	·
Guo et al. [7]	other	on-chain	✓	✓
Signorini et al.[15]	framework	on-chain	✓	✓
Khonde et al. [9]	other	on-chain	✓	✓

While there are other works that contribute to the development of ADS implementation on BT, as of current knowledge there are only three approaches that make us of blockchain meta-data and enrich it by distributing information typically generated and store at the local level to the global level main branch. Signorini et al.'s (2020) Blockchain Anomaly Detection(BAD) solution, appears to be the first approach designed to not only work with BC meta-data but also introduce protocols

meant to distribute the generated threat database at the global scale.[15] Though BAD achieved extremely less bandwidth overhead of 0.248%, only basic testing was carried out on three nodes. Hence, its application may be limited since it is not easily deployable in real networks. Guo et al. (2021) later proposed a real, and extensible ADS based scheme on a cloud computing environment, aimed to resolve scalability issues with traditional ADS due to cloud computing having a higher degree of network virtualization, flexibility and scalability.[7] Lastly, Khonde & Venugopal (2022) present a hybrid ADS using a blockchain framework to exchange signatures from one node to the other in a distributed ADS, achieving a 98.5% accuracy rate, 98.8% detection rate, and a 1.2% false alarm rate.[9] However, there is still yet to be implementation of ADS onto a full public blockchain.

3. Data collection

The implementation of a distributed ADS first involves the retrieval of *blocks* (Def. 2) and *transactions* (Def. 1) from the blockchain to build the reference model. There are two approaches to this, one consists of real time data collection that is publicly available from sources such as *etherscan.io*; the second is by querying it directly through CLI clients such as *Geth* or databases like *LevelDB* depending on which blockchain is selected. Additionally, there are a number of different datasets such as the Intrusion Detection Evaluation Dataset (CIC-IDS2017), that can be used for initial training and testing of the classifiers. The above methods depict different ways of retrieving Ethereum data specifically. Depending on the blockchain selected websites, CLI clients, and API's used to retrieve data will differ.

4. Proposed Solution

The proposed solution uses both types of methods such; signature based and anomaly based for attack detection and improving the security of the network. In the system portion, the blockchain will be used for signature transfer to the global network. This model is proposed for a distributed environment where each node is connected to the other in a distributed fashion.

4.1. *Signature based detection*

Signature based detection is performed using the standard approach where the signatures of various attacks are stored in standard datasets. There are many datasets available for intrusion detection that can be utilized to train various classifiers for identification of attacks. It is important to note that each dataset has its own set of attack signatures varying in the number and types of attacks. The Canadian Institute of Cybersecurity has published several datasets for network intrusion detection. This proposed solution utilizes four of them: CIC-IDS2017, CIC-DoS2017, CSE-CIC-IDS2018, and CIC-DDoS2019, collated into one collection to have a larger, and more varied set of NIDS sample. In this approach, the combined CIC data

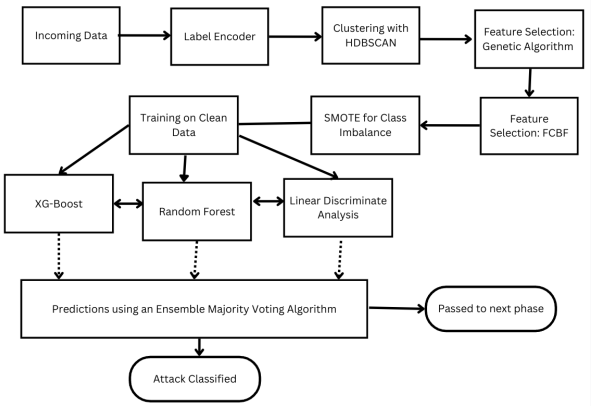

Fig. 2. Intrusion Detection System

enters the system and the goes through the preprocessing steps. This includes label encoding, clustering using a Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN), then by jointly using a Genetic Algorithm and Fast Correlation Based Filter (FCBF) for feature selection that feature space is reduced for the data. The reduced dataset is then trained on Random Forest, Linear Discriminate, and XG-Boost classifiers. All classifiers are used in an ensemble manner, and majority voting is used to determine the final prediction. If the final prediction determines that it is malicious then an alert would be sent to the global administrators and the packet would be discarded from the network. If the prediction is normal, then it is further passed onto the Anomaly detection system for further evaluation.

4.2. *Anomaly detection system*

An anomaly detection system (ADS) is a behavior-based approach for intrusion detection. The classifiers are trained using rules that represent the behavior of the packet and network. In this implementation, the classifiers are trained from the previous data in the IDS, however in a blockchain adaptation, normal behavior would be defined from all the previous block data and transaction and hash data within each block. Testing would then be assessed in real time. In an ADS since the signature of the attack is not available the behavior of the pattern will assist in recognizing malicious activities in the network.

4.2.1. *Hybrid intrusion detection system*

Hybrid IDS is developed to overcome the disadvantages of Signature-IDS (SIDS) and Anomaly Detection Systems (ADS), as it integrates Signature IDS and Anomaly Detection Systems to detect both unknown and known attacks. ADS

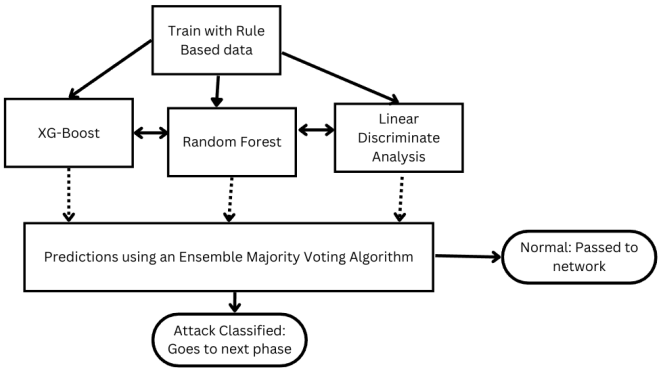

Fig. 3. Anomaly Detection System

is used to identify unseen intrusions, while SIDS is used to identify well known attacks.

4.3. *Blockchain framework for signature extraction and distribution*

The purpose of this work is to develop a hybrid IDS that is capable of protecting both the blockchain network and securely distribute signatures across the distributed network. Signature extraction is performed from the packets received from the ADS. To create the signature, a script needs to be implemented that takes the input and generates a signature equivalent to the features of the CIC-Collection data. The standard format for a signature creation is defined as such:

$$\{MACaddress, IPaddress, PublicKey, PrivateKey, Type, Port, Features\}$$

where MAC address, IP address, Public key, are responsible for signature extraction, the private key is responsible for signature extraction from a pair of public-private keys of the node, the Type is the type of attack whose signature is extracted by the node, and Port is the active communication port of an authorized node. The Features would then be the features used in the CIC-Collection dataset.

The next step would be to validate the signature created through the architecture of the blockchain, each blockchain has a different set of validators. Once that is complete the signatures are to be packaged into a block and distributed throughout the blockchain. This process of creating new blocks is called mining, and on a blockchain every block has its own unique nonce and hash. As previously shown in Fig. 1, each new block also includes the hash of the previous block, essentially creating a chain linking the blocks. Once the block is created and the mining process is complete, it is added to the network and the signatures contained in that block i.e. (previously classified signatures and new signatures added) are distributed to every user of the blockchain.

5. Results

Below are the results from performing classification on the individual XG-Boost, Random Forest, and Linear Discriminate Analysis classifiers as well as the combined voting ensemble:

Table 2. Results from indvidual and ensemble classifiers

Classifier	Accuracy	Precision	Recall	F1-Score
Random Forest	98.12%	97.55%	98.12%	97.80%
XG-Boost	98.29%	97.25%	98.29%	97.76%
LDA	83.21%	83.44%	83.21%	81.99%
Ensemble	98.31%	97.41%	98.31%	97.81%

After training and assessing the performance of individual classifiers and a voting ensemble, it is shown that the ensemble was able to slightly perform better than XG-Boost which has the second highest accuracy. Out of the three classifiers, Linear Discriminate Analysis performed the worst with an accuracy of 83.21% and F1-Score of 81.99%.

6. Limitations

The results above showed that individual performance of a Random Forest and XG-Boost Classifier is slightly less than an ensemble classifier, while the LDA classifier performed the worst out of the four. While opting for usage of a single classifier to reduce computational overhead, the use of a single classifier can often lead to overfitting. This proposed classification model is able to achieve high performance without overfitting due to [6]:

(1) It was trained on a large-sized data; the usage of more data sample can improve the generalizability of the proposed method
(2) The model proposed implements a comprehensive feature engineering method to improve the generalizability by removing irrelevant and misleading features that may cause overfitting, as well as applying SMOTE to artificially adjust class imbalance
(3) The usage of a stack ensemble method to combine the results of multiple classifiers allows the voting ensemble to have better generalizability. Combination of the single learners can reduce the variance of an estimation and prevent over-fitting

The current system proposes a method for developing a mixed hybrid detection system. At the time of writing this system was just assessed on the CIC-Collection dataset, and has not been implemented into a blockchain architecture. Thus to assess real time performance of such a system, the proposed mixed IDS/ADS learner has to be deploy into a blockchain environment through the use of HyperLedger's

technology. This will allow to further assess how the system will perform in real time detection and distribution of the new blocks to the network in the presence of new anomalous activity.

7. Conclusion

To enhance the detection of anomalous activity on the blockchain network, a hybrid IDS/ADS solution is proposed. The proposed system consisted of data preprocessing, feature engineering and utilized multiple classifiers to assess performance on the CIC-Collection data. Label encoding and HDBSCAN clustering were performed to pre-process the data. For feature engineering, a combination of a genetic algorithm and FCBF were applied to reduced the feature space. Finally SMOTE was performed to adjust for class imbalances. In assessing the performance of individual classifiers over a majority voting ensemble, it was discovered the XG-Boost had the second highest accuracy, with the highest being the voting ensemble at 98.29%. Future work of the proposed solution would be to test it on a Hyperledger environment and generate block and hashing data through the use of sources such as *etherscan.io*. Additional future work also would include adjust the hyper-parameters of the learners through Particle Swarm Optimization.

References

1. A. Alkhalifah, A. Ng, P. Watters and A. S. M. Kayes, A mechanism to detect and prevent ethereum blockchain smart contract reentrancy attacks, *Frontiers in Computer Science* **3** (02 2021) p. 598780.
2. L. Cavallaro and R. Sekar, Taint-enhanced anomaly detection, Vol. 7093 (12 2011) pp. 160–174.
3. G. Coulouris, J. Dollimore, T. Kindberg and G. Blair, *Distributed Systems: Concepts and Design*, 5th edn. (Addison-Wesley Publishing Company, USA, 2011).
4. R. Di Pietro and L. V. Mancini, *Intrusion Detection Systems*, 1 edn. (Springer Publishing Company, Incorporated, 2008).
5. M. J. Fischer, The consensus problem in unreliable distributed systems (a brief survey), in *Proceedings of the 1983 International FCT-Conference on Fundamentals of Computation Theory* (Springer-Verlag, Berlin, Heidelberg, 1983) p. 127–140.
6. B. Ghojogh and M. Crowley, The theory behind overfitting, cross validation, regularization, bagging, and boosting: Tutorial (05 2019).
7. W. Guo, L. Xie, F. Hang, Y. Lv and Z. Luo, Blockchain-based intrusion detection scheme for cloud-computing, in *ISCTT 2021; 6th International Conference on Information Science, Computer Technology and Transportation* (2021) pp. 1–4.
8. H. S. Javitz and A. Valdes, The sri ides statistical anomaly detector, *Proceedings. 1991 IEEE Computer Society Symposium on Research in Security and Privacy* (1991) 316–326.
9. S. Khonde and U. Venugopal, Hybrid intrusion detection system using blockchain framework, *EURASIP Journal on Wireless Communications and Networking* **2022** (06 2022).
10. S. Kushwaha, S. Joshi, D. Singh, M. Kaur and H.-N. Lee, Systematic review of security vulnerabilities in ethereum blockchain smart contract, *IEEE Access* **PP** (01 2022) 1–1.

11. S. Murtaza, A. Hamou-Lhadj, W. Khreich and M. Couture, Total ads: Automated software anomaly detection system (12 2014) pp. 83–88.
12. A. P. Namanya, A. J. Cullen, I. Awan and J. P. Disso, The world of malware: An overview, *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)* **0** (2018) 420–427.
13. W. Penard and T. van Werkhoven, On the secure hash algorithm family, *Cryptography in context* **0** (2008) 1–18.
14. M. Rahouti, K. Xiong and N. Ghani, Bitcoin concepts, threats, and machine-learning security solutions, *IEEE Access* **6** (2018) 67189–67205.
15. M. Signorini, M. Pontecorvi, W. Kanoun and R. Pietro, Bad: A blockchain anomaly detection solution, *IEEE Access* **8** (01 2020) 173481–173490.
16. S. E. Smaha, Haystack: an intrusion detection system, *[Proceedings 1988] Fourth Aerospace Computer Security Applications* (1988) 37–44.
17. Y. Song, A. Keromytis and S. Stolfo, Spectrogram: A mixture-of-markov-chains model for anomaly detection in web traffic. (01 2009)
18. S. Tanwar, Q. Bhatia, P. Patel, A. Kumari, P. K. Singh and W.-C. Hong, Machine learning adoption in blockchain-based smart applications: The challenges, and a way forward, *IEEE Access* **8** (2020) 474–488.
19. S. Thompson, P. L. Seijas and D. Adams, Scripting smart contracts for distributed ledger technology, none (December, 2016).
20. M. Vukolić, The quest for scalable blockchain fabric: Proof-of-work vs. bft replication (05 2016) pp. 112–125.
21. X. Wang, J. He, Z. Xie, G. Zhao and S. C. Cheung, Contractguard: Defend ethereum smart contracts with embedded intrusion detection, *IEEE Transactions on Services Computing* **13** (2020) 314–328.
22. R. Wattenhofer, *The Science of the Blockchain*, 1st edn. (CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 2016).
23. J. Xu, Are blockchains immune to all malicious attacks?, *Financial Innovation* **2** (12 2016).
24. Z. Zheng, S. Xie, H. Dai, X. Chen and H. Wang, An overview of blockchain technology: Architecture, consensus, and future trends, *2017 IEEE International Congress on Big Data (BigData Congress)* **0** (2017) 557–564.
25. A. Zohar, Bitcoin: Under the hood, *Commun. ACM* **58** (aug 2015) p. 104–113.