BLOCKCHAIN-BASED ARCHITECTURE FOR SECURED CYBERATTACK SIGNATURES AND FEATURES DISTRIBUTION

Oluwaseyi Julius Ajayi

Department of Electrical Engineering

Mentor: Prof. Tarek Saadawi

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of the Doctor of Philosophy

Tarek Saadawi, Chair of Examining Committee	Date
Ardie D. Walser, Associate Dean for Academic Affairs	Date

EXAMINING COMMITTEE

Myung Lee, Professor of Electrical Engineering, The City College of New York

Akira Kawaguchi, Professor of Computer Science, The City College of New York

Abbe Mowshowitz, Professor of Computer Science, The City College of New York

Masato Tsuru, Professor of Computer Science and Electronics, Kyushu Institute of Technology, Fukuoka, Japan

Kenichi Kourai, Professor of Creative Informatics, Kyushu Institute of Technology, Fukuoka, Japan

DEDICATION

This work is dedicated to God Almighty, the maker of all things. I want to also dedicate the work to my wife and children.

ACKNOWLEDGMENTS

I want to, first, appreciate God Almighty for the successful completion of this milestone. My sincere appreciation goes to my mentor: Prof Tarek Saadawi, for his role in actualizing this dream. I want to thank the City University of New York, City College for the CUNY research fellowship and National Science Foundation (NSF), under research grant "NeTS-JUNO2: Resilient Edge Cloud Networked Design" for the researching funding. I want to appreciate all my Ph.D. Defense committee members (both Internal and External) and the Graduate Affairs Grove School of Engineering, City College of New York. My special thanks go to Ms. Belkys Bodre for her continuous support and Assistance. I also want to appreciate the other students in CINT Lab; Huseyn, Sach, Meryem, and Xiang. I want to appreciate the past and current students that worked with me; Dr. Obinna Igbe, Melvin, Chantelle, Carlos, Stefan, and Joel.

My profound gratitude goes to my wife, Dr. Oluwaseun Ajayi, for her support, perseverance, and encouragement during my program. I want to thank my daughter, Oluwafikayomi A. Ajayi, for seeking daddy's attention, especially when busy. I also want to appreciate every member of my extended family.

Thank you all. May God continue to bless you.

OUTLINES

Lists of Figures

Figure 18: The pictorial representation of the e-health application

Figure 19: The Building blocks of the e-health application

Lists of Tables

ABSTRACT

One effective way of detecting malicious traffic in computer networks is intrusion detection systems (IDS). Despite the increased accuracy of IDSs, distributed or coordinated attacks can still go undetected because of the single vantage point of the IDSs. Due to this reason, there is a need for attack characteristics' exchange among different IDS nodes. Another reason for IDS coordination is that a zero-day attack (an attack without a known signature) experienced in organizations located in different regions is not the same. Collaborative efforts of the participating IDS nodes can stop more attack threats if IDS nodes exchange these attack characteristics among each other. Researchers proposed a cooperative intrusion detection system (CoIDS) to share these attack characteristics effectively. Although this solution enhanced IDS node's ability to respond to attacks previously identified by cooperating IDSs, malicious activities such as fake data injection, data manipulation or deletion, data integrity, and consistency are problems threatening this approach.

In this dissertation, we develop a blockchain-based solution that ensures the integrity and consistency of attack characteristics shared in a cooperative intrusion detection system. The developed architecture achieves this result by continuously monitoring blockchain nodes' behavior to detect and prevent malicious activities from both outsider and insider threats. Apart from this, the architecture facilitates scalable attack characteristics' exchange among IDS nodes and ensures heterogeneous IDS participation. It is also robust to public IDS nodes joining and leaving the network. The security analysis result shows that the architecture can detect and prevent malicious activities from both outsider and insider attackers, while performance analysis shows scalability with low latency.

CHAPTER ONE

INTRODUCTION

The development of information technologies and the trend of integrating cyber, physical, and social (CPS) systems into a highly unified information society are becoming increasingly apparent, especially with the recent proliferation of smart devices and the Internet of Things (IoT) [1]. Such an information society generates and shares a massive amount of data anytime, and the data's security should be under the control of its owner [2-3]. An increasing amount of personal data, including location information, web-searching behavior, user calls, and user preference, is silently collected by the built-in sensors inside the products from those big companies, bringing colossal privacy leakage risk for the data owners [4-5]. Given the tremendous increase in information technology data, companies have searched for more secure ways of exchanging data among users. Companies store data in their on-premises data centers to manage data security; however, it is challenging to manage this data locally, especially now that they deal with exabyte scale data. To exchange data, data owners often rely on internet providers for the security of in-transit data. The reliance on internet providers makes such data exposed to third-party and confined the data transfer to the provider's communication channel, which in most cases has security vulnerabilities, thus, making data security out of the control of their owners. The inability to effectively manage in-transit data security makes it very challenging for an individual to control the collected data's risks [6].

The increasing popularity of the internet has made data owners turn to cloud solutions for secured data storage and exchange. Thus, cloud storage is gaining popularity not only among enterprises but also among small organizations and individuals. For example, OpenStack [7], an open-source cloud computing platform, is becoming the de-facto standard for Infrastructure-as-a-Service clouds, as it is backed by technology giants like IBM, Dell, Cisco, which sell and support OpenStack-based solutions. Also, Amazon Web Service (AWS), Google, and Microsoft Azure are becoming forces to reckon with because of the security, durability, availability, and scalability of data. Consequently, companies rely on cloud storage for their data because it is more secure and easily accessible; however, the caveat is that the internet-based stored data's attack surface has increased tremendously. Several kinds of research put

forward ways of securing both data at rest (stored data) and data in transit. For instance, the authors in [8] proposed firewall, user authentication, and data encryption to control data access in a computer network. Although a network built around user authentication, firewall, and data encryption systems has been able to secure companies and individuals from cyber-attacks, malicious intruders continuously find ways to subvert these protection systems to attack networks and gain unauthorized access to stored data.

Further research described intrusion detection systems (IDS) as an additional wall to identify malicious intruders in computer networks and devices connected to the Internet [9-11]. Intrusion detection can provide two main functions: a) *Information Recording:* An IDS can monitor the target objects and record information locally, then sends the collected data to other facilities for analysis, like central event management. b) *Alert Generation:* The main task of an IDS is to generate alerts to inform security administrators of significant identified anomalies. False alarm rates are an essential measurement to decide whether an IDS is effective or not [12]. An IDS can generally be classified based on their locations: Host-based intrusion detection system (HIDS) and network-based intrusion detection system (NIDS) [11] or their detection approaches: signature-based and anomaly-based [10] (Figure 1).

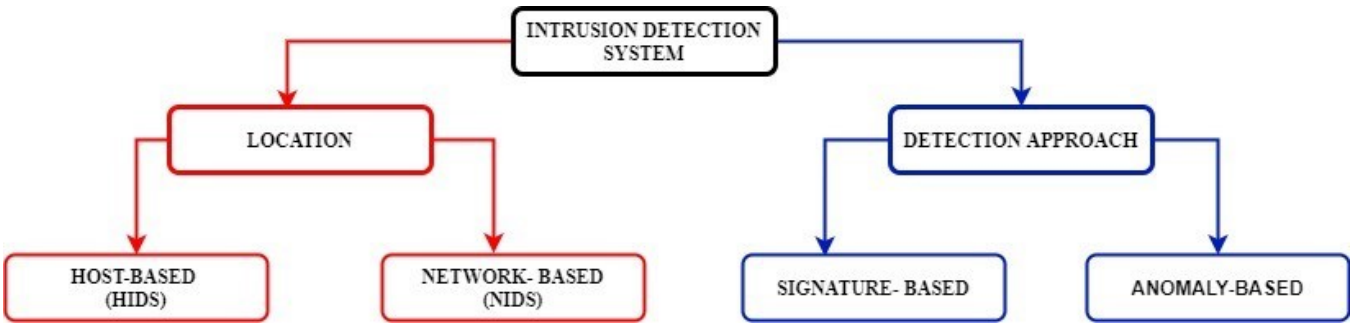

Fig 1. Classification of Intrusion Detection Systems (IDS)

A HIDS analyzes the traffic to and from a specific host in which the intrusion detection software resides. It is a system that is installed on the individual host to detect an intrusion or misuse and responds by logging the activity and notifying the designated authority[11]. A host-based system can monitor essential system files, process behaviors, unusual resource utilization, unauthorized access, any attempt to

overwrite files, and so on. On the other hand, a NIDS is often a stand-alone hardware appliance that includes network detection capabilities. The NIDS, located anywhere within the network, monitors network traffic and detects any malicious activities by identifying a suspicious pattern in the incoming packets[11]. It captures network packets from the network traffic. The captured packet's header is analyzed based on the various parameters (known as features) to detect malicious activities. It can be set up in the network backbone, server, switches, and gateways. HIDS and NIDS can either be signature-based or anomaly-based IDS.

A Signature-based IDS detects intrusion by comparing an incoming traffic with a set of predefined rules. The technique looks for a specific behavior pattern, which is already known as an attack [13]. All the negative patterns and behaviors which are identified as attacks are stored in the IDS signature database. This signature database is continuously updated and used for attack detection. This technique's limitation is that it cannot detect a zero-day attack (attacks for which there are no signatures), as a signature is not yet available[13]. The trade-off is that a signature-based IDS can achieve 100% accuracy if the attack rules are correctly set. In contrast, an anomaly-based IDS detects incoming traffic patterns that do not conform to the expected regular traffic pattern. In anomaly-based IDS, network traffic or host OS behavior is analyzed based on various parameters and compared to an expected behavior [10]. If incoming traffic patterns do not conform to the expected regular traffic, the system raises alarms. Although the anomaly-based IDS can detect new zero-day attacks, unlike the signature-based IDS, its accuracy level is usually below that of the signature-based IDS[11]. Signature-based and anomaly-based IDSs can be located in a host as in HIDS or the network as in NIDS.

Intrusion Detection Systems proved to help identify malicious activities in computer networks and IoT devices; however, its single vantage point limits its capability in detecting coordinated or distributive attacks. Consequently, some attacks can go undetected or not detected on time. As a result of attack escape, IDS nodes need to exchange attack information to detect new or recurring attacks promptly. Another reason for attack information sharing is that a zero-day attack experienced by organizations located in

different regions is not the same. For instance, a zero-day attack experienced by an organization located, say in London, the United Kingdom, might not be the same as that experienced by another organization located, say in New York, the United States, or another company located in the same region. Therefore, if IDS nodes can exchange this threat information, more malicious activities can be stopped by coordinating efforts of the participating IDSs.

1.1 Cooperative Intrusion Detection System (CoIDS)

A cooperative intrusion detection system (CoIDS) was proposed to enhance the detection coverage of single IDS [14-16]. In the CoIDS, IDS nodes exchange attack signatures with neighboring IDS nodes with the view of promptly detecting an attack that other IDS nodes have previously detected. CoIDS is a framework that collects attack signatures from IDS nodes and disseminates this information to other neighboring IDS. Often, CoIDS engages a centralized process, such as a cloud database, in data dissemination[15]. Companies adopted cooperative intrusion detection system because of the better performance; however, some of the major problems threatening the approach are:

- *Data manipulation:* Malicious intruders can hack the database and alter the data that is being exchanged even if it is encrypted.
- *Data deletion:* Stored data can be deleted from the database by a malicious insider or outsider if the activities are not monitored.
- *Fake data injection to the database*: When data manipulation is not readily achievable, a malicious intruder can inject fake data into the database if hacked, thereby compromising the shared data integrity.
- It might be challenging to guarantee the shared data's consistency due to a compromised medium of exchange.
- A centralized platform utilized by CoIDS can make the network susceptible to single-point-of-failure attacks.

- There is a need to trust a third party that manages the database's activities, making the network vulnerable to man-in-the-middle attacks.

1.2 Statement of the problem

As reported in [17], achieving attack information distribution in the existing CoIDS solution is divided mainly into four stages (Figure 2), and cyber attackers attempt to compromise the shared information by targeting the storage and distribution stage.

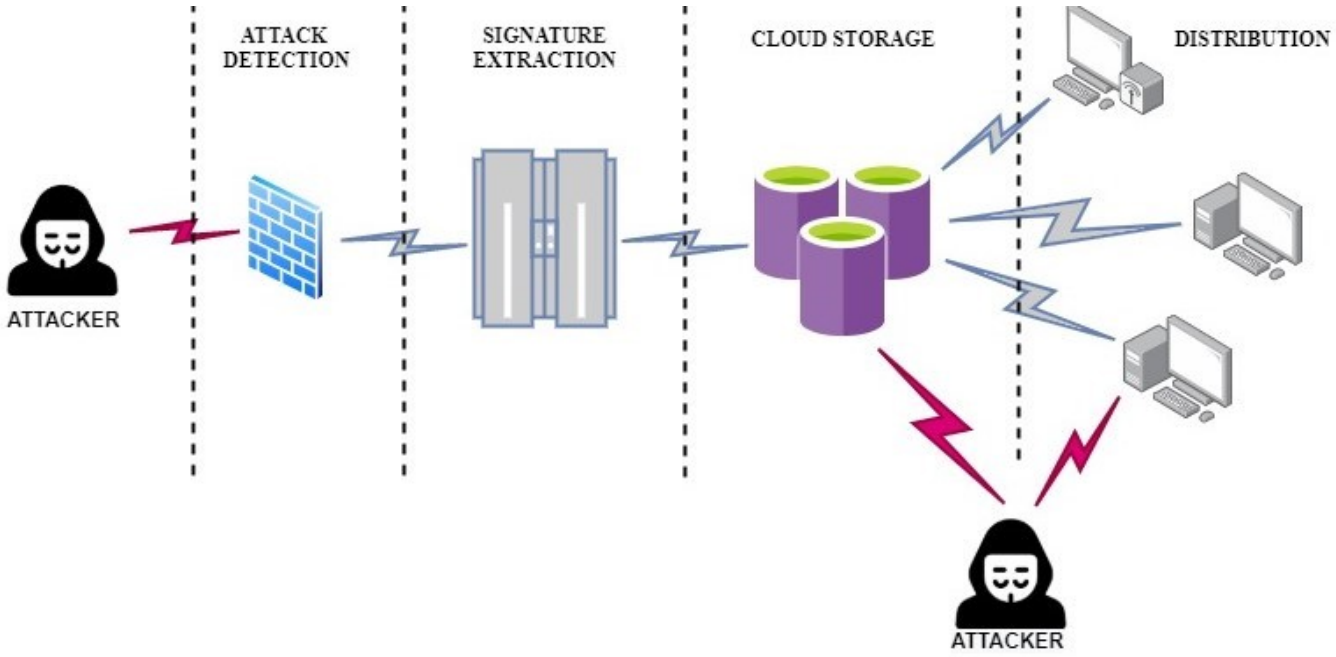

Fig. 2. Cyber-attack targets of existing CoIDS

Diverse research put forward methods to secure both stored and in-transit data [14,15, 18-20]. However, most of these existing researches focus on utilizing a centralized approach (which makes the network vulnerable to single-point-of-failure and man-in-the-middle attacks [14,15,18]) or uses a decentralized approach in which the integrity and consistency of the shared data cannot be guaranteed [19-20]. Data users believe that stored or in-transit data is secured as long as it is encrypted. Although encryption ensures such data's confidentiality, it is not tamperproof as it does not guarantee such data's consistency and integrity. Several research pieces proposed ways of securing data integrity [21]. However, most of the methods are not practical, especially for big data. For instance, authors in [21] proposed a message

authentication code (MAC) to secure data integrity. In this method, a data file is downloaded, and the hash value is checked. This method detects accidental and intentional changes in the data; however, downloading and calculating large files' MAC is overwhelming and time-consuming. Another method proposed in [21] secures cloud data integrity by computing the hash values of every data in the cloud using a hash tree. This solution is lighter than the first method. However, it is also not practical because computing the hash values of massive data requires more computational power, and it is time-consuming. The authors in [22] utilize a third party to coordinate the database's activities. In this method, a person with skills and experience carries all auditing processes, such as checking data integrity. The approach's problem is the need to trust the third party, which exposes data to a man-in-the-middle attack or the network to a single-point-of-failure attack.

In this work, we propose a solution that leverages blockchain technology abilities to store, detect and prevent malicious activities on data (e.g., cyberattack features and signatures, Patient health records) and distribute them to nodes in different regions in real-time. The approach extracts cyberattack features or signatures from an IDS node, invokes server-side authentication for transaction owners, verifies the data's consistency and integrity, presents it in a standard format, and stores it into an immutable decentralized database. The solution securely distributes the attack features or signature among participating IDS nodes that subscribe to the blockchain network in real-time to achieve cooperative intrusion detection systems (Figure 3). We define attack features as characteristics of attacks retrieve from attack traffic detected by an IDS. On the other hand, attack signatures are predefined rules that detect a specific attack, and it is obtained from signature-based IDS. We also define a transaction as any information (e.g., an attack signature or features) that a node submits to the blockchain network.

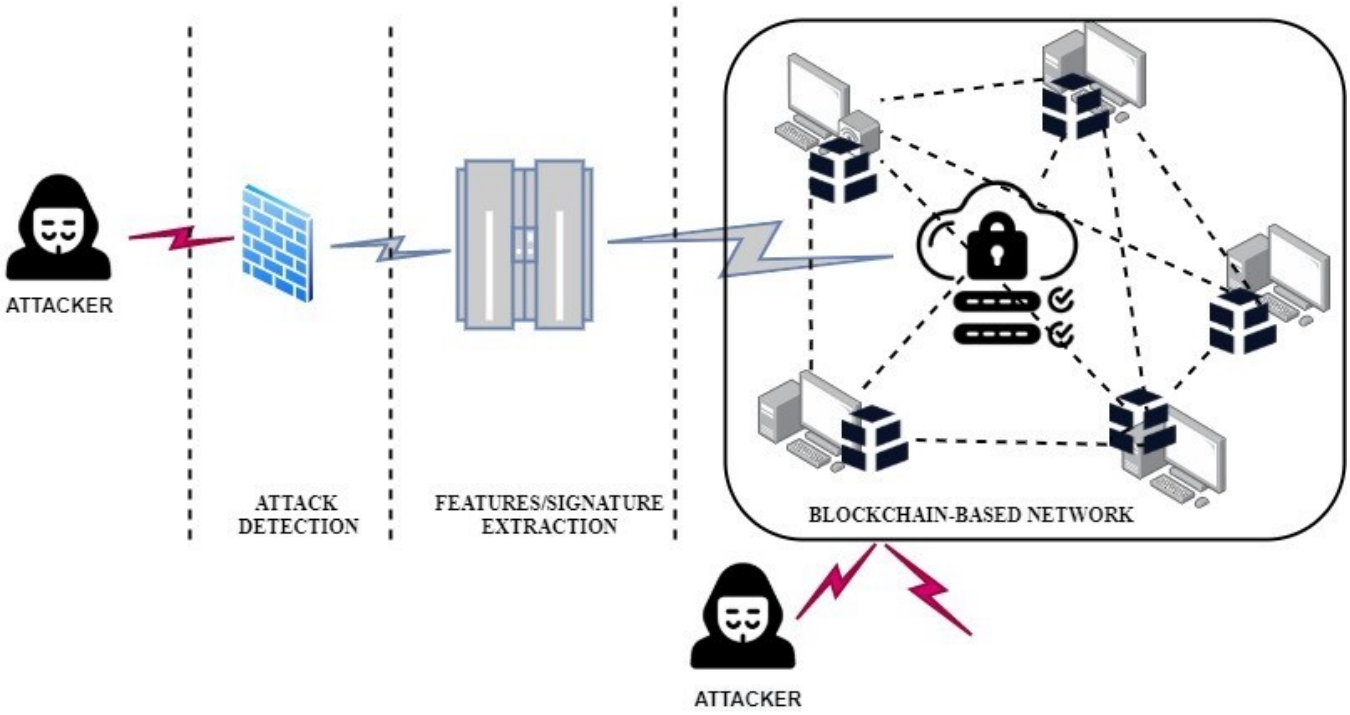

Fig. 3. The proposed blockchain-based solution for CoIDS.

1.3 Contributions of the work

The contributions of the work can be summarized as follows:

- We propose a private-public blockchain-based architecture that facilitates scalable and secured attack signature exchange among IDS nodes in computer networks. The architecture retrieves attack signature from any signature-based IDS, e.g., SNORT[23], then securely distribute it among other participating signature-based IDS nodes, e.g., Bro[65], Suricata[66], in a computer network for cooperative intrusion detection systems.
- We propose a blockchain-based architecture that can facilitate a scalable and secured exchange of attack features among IDS nodes. The architecture retrieves attack features from any anomaly-based IDS (e.g., Dendritic Cell Algorithm (DCA)[9]) using a developed script, then securely distribute it among other participating IDS.
- The developed architecture detects and prevents malicious activities on either stored or in-transit data from outsider threat actors. The architecture detects and prevents a public node from

compromising the stored attack information by leveraging the server-side authentication code executed by the smart contract.

- The proposed architecture monitors the activities of blockchain nodes to detect compromised insider nodes. The architecture identifies and thwarts the malicious activities of compromised authorized blockchain nodes by continuously examining the individual node's behavior. We evaluate the detection ability of the proposed architecture based on attacks that fall under three categories: *(a)Those attacks targeting the Blockchain to bring it down (b) the attacks that attempt to inject fake data into the database (c) The attacks that attempt to hijack or retrieve unauthorized data*.
- The proposed architecture provides strong server-side authentication and verification procedures for the transaction owner and submitted transactions. The authentication and verification procedures ensure that only non-malicious authorized nodes can submit a transaction and verify the submitted features or signatures' integrity and consistency. The architecture converts a successfully verified transaction into a standard format that different IDSs can easily understand; thus, the architecture encourages heterogeneous IDS node participation. The smart contract invokes a code that converts attack signatures to standard format based on the mandatory variables. We defined a format for the features and ensured that the verification code compares an incoming transaction with the format before converting it to a standard or generalized format.
- The architecture permanently stores the verified transaction in a tamper-proof, immutable, distributive blockchain network. Contrary to the centralized or other distributed storage system that is vulnerable to malicious activities, transactions in a blockchain network are permanently stored, and it is impossible to manipulate or delete. Access to the stored transaction is shared among the authorized participating IDS nodes in real-time.
- The proposed architecture is robust to public IDS nodes joining and leaving the network in real-time. The architecture grants permissionless access to public IDS nodes to join the blockchain network and retrieve stored attack features or signatures in real-time without posing any security

concern. The architecture can also restrict transaction retrieval access to specific nodes if the information is not meant for every node.

- We also proposed an application of the developed architecture in the healthcare system. Using the same concept, we described how the architecture could securely share Electronic Health Records (EHRs) between two or more Healthcare Systems. We further described how we implement the transaction access management to restrict transaction access to a requesting node.

CHAPTER TWO

BLOCKCHAIN TECHNOLOGY

2.1 Description of Blockchain

Blockchain technology has recently received significant interest from various international organizations, industries, and institutions and has even been expressed by some researchers as a more robust technology than the Internet [24]. Before the popularity of blockchain applications, there was a need to trust a third party for digital assets' security and privacy. With the invention of blockchain, people no longer need a third-party agent to provide security and verification in their product or service transfer operations. The "trust protocol" created by blockchain provides a reliable, transparent, and accountable environment [25]. First introduced as the technology behind bitcoin in 2008 [26], blockchain was implemented to solve the double-spending problem in a cryptocurrency called bitcoin. Blockchain technology is a new type of database that non-trust parties can directly share without requiring a central administration, unlike SQL or other databases. It is a distributed, decentralized, transparent, chronological, and append-only public ledger that records all network transactions. Every participant in the blockchain network is called a node. The data in the blockchain is known as a transaction, and it is divided into blocks. Each block is dependent on the previous one (parent block). Every block stores some metadata and hash value of previous blocks; hence, every block has a parent block[26].

The network's algorithm ensures that each transaction is unique. Each block of transactions in the public ledger is verified by consensus (an agreement among participating nodes) of most of the system's participants. The consensus is achieved by a proof-of-work protocol in a mining process. A proof-of-work protocol is a mathematical algorithm that is difficult to produce, but it is easy to verify. Once the transaction is verified, it is impossible to mutate/erase the records (i.e., none of the users, even the administrators, can change or delete the existing chain of blocks) [26]. To determine with high probability that a transaction is permanently included in the blockchain, one must wait for several blocks to be mined after the first inclusion and refer to the block, including the transaction of interest as an ancestor. Each of

these subsequent blocks is called confirmations, and when sufficient confirmations occurred after the transaction block inclusion, then the transaction is considered committed, i.e., believed to remain in the blockchain forever with high probability. For instance, the current version of Ethereum requires 11 blocks after the transaction inclusion for the transaction to be committed [27].

Blockchain is revolutionizing the digital world by enabling a distributed consensus where every online transaction, past and present, involving digital assets can be verified at any time in the future. It does this without compromising the privacy of the digital assets and parties involved. The distributed consensus and anonymity are two essential characteristics of blockchain technology[26]. Blockchain is broadly divided into two: public and private blockchain [12]. A public blockchain, also known as a permissionless blockchain, is where anybody can join the network to be a verifier without obtaining any prior permission to perform such network tasks. There is no centralized authority, or no party has more power than the rest. Here, everyone is opened to join or leave as they wish. The blockchain is publicly opened, and everyone has the right to validate a transaction. Since anybody can join, specific types of incentive mechanisms are necessary for verifiers to participate. For instance, the bitcoin blockchain's validators get bitcoins (BTC) in transaction fees, and the new bitcoins are generated for the effort they put into solving the proof-of-work challenge. Blockchain has the advantage that it can accommodate both anonymous and pseudo-anonymous actors. Bitcoin and Ethereum are examples of permissionless blockchains[28].

On the other hand, private blockchains are permissioned blockchains where special permission is needed from an authority to become a verifier in the system. Permissioned blockchains are intended to be purpose-built and can thus be created to maintain compatibility with existing applications (financial or otherwise). A Permissioned blockchain is known to be faster, more energy-efficient, and scalable than a permissionless blockchain. It can be easily implementable than a permissionless blockchain. In a permissionless blockchain, the data is stored on every blockchain node, and all nodes verify all transactions. In a permissioned blockchain, not everyone has equal rights to validate transactions. Thus, only a few nodes are given certain privileges of validating the transactions. The rest may validate, but

these selected participants must reach a consensus before the implementation. However, it is much easier for users to collaborate and alter the rules or revert transactions because of the smaller number of participants; hence, 51% attack threatens private blockchain [28]. That is why only trusted parties should be permitted to act as verifiers. Examples of permissioned blockchains include Eris, Hyperledger, and Ripple.

2.2 Basic Terminologies in Blockchain Technology

Hash Function: This a function that converts letters and numbers input of any length into an encrypted output of a fixed length. In a blockchain, hashing requires processing the data from a block through a mathematical function, resulting in an output of a fixed length. Using a fixed-length output increases security since anyone trying to decipher the hash will not be able to tell the length of the input by merely looking at the length of the output. The function used to generate the hash is deterministic, meaning that it will produce the same result each time the same input is used [71].

Nonce: A nonce is an abbreviation for " number only in used once." It is a string of numbers added to a hashed or encrypted block in a blockchain, that when rehashed, meets the difficulty level of restriction. The nonce is the number a blockchain miner needs to discover before successfully solving a block in the blockchain. To determine the correct nonce requires a significant amount of trial-and-error, as it is a random number. A miner must guess a nonce, append it to the current header, rehash the value, and compare this to the target hash. If the hash meets the target restriction, the miner has successfully validated the block [72].

Target Hash: A target hash is a number that a hashed block header must be less than or equal to for a new block to be awarded. The target hash is used in determining the difficulty in a blockchain and can be adjusted to ensure that blocks are processed efficiently. The difficulty or target hash is calculated based on the timestamp on the previous block header. Each miner calculates the difficulty of the same block independently since everyone uses the same algorithm. Thus, everyone gets the same results. The timestamp of the target hash is adjusted based on the previous block's timestamp [73].

Merkle Tree: A Merkle tree involves taking a large amount of data and making it more manageable to process. In a most general sense, a Merkle tree is a way of hashing a large number of "chunks" of data together, which relies on splitting the chunks into buckets, where each bucket contains only a few chunks[29]. It is implemented by taking each bucket's hash and repeating the same process, continuing to do so until the total number of hashes becomes only one, the root hash, and place it in the block header. It represents the summary of all transaction data. The Merkle tree is helpful because it allows users to verify a specific transaction without downloading the whole blockchain. With Merkle trees, it is possible to build blockchain nodes that run on all computers and laptops, large and small smartphones, and even the Internet of things devices [29].

Mining: Mining adds blocks of transactions to the massive distributed public ledger of actual transactions, known as the blockchain. Mining involves validating new transactions and recording on the distributive ledger. To validate a transaction, the miners verify that the transaction is legal (neither malicious nor double-spend) [75].

Consensus Protocol: Consensus involves agreeing on the ordering of the validated transactions. Each time a transaction is made, it is broadcasted to the entire network. Upon hearing the broadcasts, miners take a bunch of transactions, validate that they are legitimate, and put them into a block. However, miners hear about different transactions at different times (due to latency issues). Furthermore, they may simply choose different transactions to include in their block base on transaction fees. So essentially, each miner is building his block, and this block may be completely different from the rest of the miners in the network. Miners do not need to build the same global block; thus, they can build their block consisting of entirely different transactions. Moreover, the participants will come to a consensus on which block is included next. Only one block can be added at a time, and the miners need to agree upon which one, hence, the introduction of consensus protocols [76]. There are different types of consensus protocols; some of them are explained below

Proof-of-Work (PoW): Proof-of-Work is essentially a puzzle that consists of a challenge to which proof must be found. This concept has existed before Bitcoin and virtual cryptocurrencies, and it has been used in spam and DDoS prevention. In PoW, transactions are grouped into a block, validated, and confirmed by the miners. The miners are required to solve a challenge by computing cryptographic hashes. They achieve this by making trial and error computations until a consensus is reached. PoW works because all miners compete to find the nonce, so the block will hash to a target value determined by the mining difficulty when the nonce is combined with the proposed block. A proof-of-work puzzle's characteristics are: Computationally challenging to compute and easy to verify [26].

Proof of Stake (PoS): Proof of Stake refers to a method to assign priority in hash calculations per virtual currency holding ratio[77]. In PoS, mining new blocks depend on who holds the highest amount of cryptocurrency, in which a deterministic algorithm selects nodes according to the number of coins each one has. Hence, instead of investing in expensive computational power to mine blocks, miners invest in the system's currencies. As a result, a miner's likelihood of being chosen to create a block depends on the fraction of coins the miner owns in the system [78]. For example, someone holding 1% of the Bitcoin can mine 1% of the Proof of stake blocks. Proof of stake is based on the idea that a dishonest act committed by a node holding a large number of virtual currency reduces the currency's reliability and value. This fact works as an incentive for any participant to avoid dishonest acts.

Proof of Importance (PoI): A Proof of importance (POI) is a system used to determine which users are eligible to perform the calculations necessary to add a new block of data to a blockchain and receive the associated payment. A proof of importance algorithm prioritizes miners based on the number of transactions in the corresponding cryptocurrency that they perform. The more transactions are made to and from an entity's cryptocurrency wallet, the higher that entity's chances of being given mining projects are. It uses transaction amounts and balances of individual nodes as indicators, calculates each node's significance, and assigns priority in hash calculations to more significant nodes. Clustering makes it possible to detect nodes that are likely to commit unlawful transactions [79].

Practical Byzantine Fault Tolerance (PBFT): PBFT is an algorithm for solving a Byzantine Fault resulting from a failure in building a consensus caused by the Byzantine general problem. It was challenging to put a theoretical algorithm to practical use due to the enormous computation requirement [80]. PBFT works in an asynchronous environment that might contain byzantine faults such as the internet. The nodes in PBFT are divided into primary and backups. In PBFT, a client sends a request to the primary. The primary assigns the request a sequence number and multicasts a signed pre-prepared message to other backups. The pre-prepared contains the view and sequence number. If the backups did not already accept a pre-prepared message for the same view and sequence numbers, they would accept the pre-prepared. After accepting the pre-prepare, a backup broadcast the signed prepare message. PBFT uses a timeout mechanism to deal with fault primaries [80].

Incentives for Miners: Incorporating a block into a blockchain requires energy from the miners. To verify a transaction and create a block, miners use computation power, which has its cost. Only the miner who comes up with the proof for the challenge gets the award for the current block. The rest gets nothing and must start working on a new block, and so on. So, for every block in the blockchain, only one miner gets awarded. For instance, in the bitcoin blockchain, the miners get bitcoin (BTC) as incentives for incorporating a transaction block into the chain, while Ethereum miners get Ethers (ETH) as incentives for validating a block[26]. The more transactions are in the block, the more the transaction fees are collected if the block is published successfully [75].

Gas Limit: The term gas limit refers to the maximum price a cryptocurrency user is willing to pay when sending a transaction or performing a smart contract function. In Ethereum, gas fees, measured in gwei, are payments made by users to compensate for the computing energy required to process and validate transactions. The gas limit refers to the maximum amount of gas (or energy) that a sender is willing to spend on a particular transaction. The higher the gas price the sender is willing to pay, the more critical the transaction since the miner's reward will be higher [27].

Smart Contract: The fundamental purpose behind the inception of Blockchain smart contracts was to allow the performance of credible transactions without involving third parties. A smart contract is a code stored on the blockchain and automatically executed when predetermined terms and conditions are met. A smart contract is a computer program that executes agreements established between at least two parties, causing certain actions to happen when a series of specific conditions are met. Thus, when such previously programmed conditions occur, the smart contract automatically executes the corresponding clause. In this way, the code may translate into legal terms that control physical or digital objects through an executable program [30].

Simply put, a smart contract works a lot like a vending machine. A smart contract can work on its own, and it can also be implemented along with any number of other smart contracts. Smart contract conditions are based on data that depends on external services that take data from the real world and store them into the blockchain (or vice versa). Their life on the blockchain network ensures they cannot be altered, and thus they provide the guarantee of trust that previously required elaborate controls and audit processes conducted by trusted third parties and arbitrators. Smart contracts contain complex and detailed instructions that automate processes that otherwise require oversight by centralized authorities. They eliminate costly duplication of manual effort [31]. In information exchange, smart contracts allow for creating dynamic and scalable rules that can be used to exchange patient information securely. They are suitable for reorienting processes that add efficiency to systems by the removal of human agents. The inclusion of smart contracts in the information exchange adds an extra layer of security that is also efficient to healthcare delivery, such as removing trusted third parties, reducing financial costs, inefficiencies, and wastage [31]. They can be set up in a way that will be dependent on one another. A smart contract contains three integral parts: Signatories, two or more parties agreeing on using the smart contract, subject of agreement (i.e., the agreed contract), and the specific term, including the rules, rewards, and punishments associated with the subject of the agreement[31].

Fork: A fork is defined as what happens when a blockchain diverges into two potential paths or a change to the blockchain protocol. (i.e., a situation that occurs when two or more blocks have the same block height). Ideally, a unanimous consensus among the blockchain nodes results in a single blockchain containing verified data(transactions) that the network asserts to be correct. However, many times, the network nodes cannot reach a unanimous consensus regarding the blockchain's future state. This event leads to forks in which the ideal 'single' chain of blocks is split into two or more chains, which are all valid. Blockchain forks can occur due to new functionality, fixing security issues, or reversing transactions [27]. Blockchain forks are mainly divided into two: Hard and soft forks[39].

Since its inception, blockchain has applications in diverse areas, e.g., health system [32-38], data integrity security [39], intrusion detection system [40-42], and others. Here, we explore its application as an alternative and fast method of securely distributing cyberattack features and signatures among different participating nodes. Each participating node could represent the gateway of different companies. The proposed architecture could serve as a medium where different companies form a consortium to exchange attack signatures and features to detect distributed or coordinated attacks promptly. Furthermore, we applied the developed architecture to patient health records exchange across different healthcare systems. Given that healthcare systems embrace interoperability to deliver better healthcare service to patients, our architecture will be practical because it detects and prevents malicious activities on data, thus ensuring the stored and shared data integrity and consistency.

CHAPTER THREE

BACKGROUND AND RELATED WORKS

Several kinds of research have proposed solutions to address secure information sharing among IDS nodes in CoIDS over the years. Although some of these methods are practical, they expose the system to other vulnerabilities that cyberattacks can exploit. Some of these proposed solutions are discussed below;

3.1 Cooperative intrusion detection

The authors in [43] proposed a prototype Distributed Intrusion Detection System (DIDS). The proposed system combines distributed monitoring and data reduction with centralized analysis to monitor a heterogeneous network of computers. They considered how to track a user moving across the network with a new user-id on each computer. The result showed that the prototype demonstrated the viability of distributed architecture in solving the network-user identification problem. This solution is practical; however, with the DIDS director responsible for all evaluation, the system is susceptible to single-point-of-failure or man-in-the-middle attacks. Another research in [44] proposed DOMINO (Distributed Overlay for Monitoring Internet Outbreaks). DOMINO is an architecture for a distributed intrusion detection system that fosters collaboration among heterogeneous nodes organized as an overlay network. In their system, they used active-sink nodes that respond to and measure connections to unused IP addresses. The configuration enables efficient detection of attacks from spoofed IP sources, reduces false positives, enables attack classification and production of timely blacklists. They evaluated DOMINO's capabilities and performance using a broad set of intrusion logs collected from different providers across the Internet. The result showed that DOMINO helps discriminate between types of attacks by examining the payload data. Although their system showed a good result, an attacker can hack the database that manages activities, manipulates stored data, or inject fake data into the database.

In [45], the authors proposed a cooperative intrusion detection system (CoIDS), which uses a cooperative approach for intrusion detection. In their method, individual intrusion detection components work

cooperatively to perform concerted detection. The result showed that the system is efficient and effective in preventing viruses from spreading in a chain way. However, with the introduction of an intrusion detection manager (IDM), who maintains and updates cooperative protocols, rules, and logs, there is a need to trust IDM, which may expose the network to man-in-the-middle or single-point-of-failure attacks. In another research put forward in [46], the authors proposed a cooperative intrusion detection framework in cloud computing to reduce the impact of denial-of-service attacks (DoS) and distributed denial of service attacks (DDoS). Each IDS has a cooperative agent that computes and determines whether to accept the alerts sent from other IDS. The result showed that the proposed system only increases little computation effort than pure snort-based IDS but prevents the system from a single point of failure attack. However, the system failed to address a situation when each cooperative agent uses different IDSs. The system is also susceptible to malicious intruder activities such as data hijacking via the medium of transmission.

3.2 Blockchain Application in Intrusion Detection System

The authors in [47] present a Collaborative Blockchained Signature-based Intrusion Detection **(**CBSigIDS). CBSigIDS is a generic framework of collaborative blockchain signature-IDS. This framework utilized blockchains to update a trusted signature database for different IDS nodes in a collaborative network incrementally. The experiment investigated the performance of CBSigIDS against adversarial scenarios like worm and flooding attacks in simulated collaborative intrusion detection systems or networks (CIDN). In the evaluation, they compared the results from simulated CIDN against real CIDN. The result showed that blockchain technology could enhance the robustness and effectiveness of signature-based IDSs under adversarial scenarios via building a trusted signature database. Although their system works like our proposed architecture, the approach does not incorporate anomaly-based attack detection and distribution. Also, the authors failed to address how their architecture detects malicious or compromised blockchain nodes. In [48], the authors proposed a SectNet, an architecture that can secure data storage, computing, and sharing in the large-scale Internet environment. The architecture aimed at

more secure cyberspace with actual big data and enhanced AI with plenty of data sources. Their architecture integrates 1) blockchain-based data sharing with ownership guarantee, 2) AI-based secure computing platform, and 3) a trusted-value exchange platform. The performance analysis evaluated the vulnerability when suffering from notorious network attacks such as the DDoS attacks and the revenue for contributors who provide Blockchain's security rules. The result showed that the SectNet significantly reduced DDoS attack's impact due to the sharing of the security rule sets by every internet user. The contributor's revenue will also increase at a higher rate if the shared security rules are of higher quality, especially after the real market's quality effect is formed. Based on the analysis, the work is specific to DDoS attacks, and it does not address how it can detect compromised insider nodes. Also, the authors failed to explain how stored information is verified.

Further research put forward by [49] presents a trust chain that mitigates attacks that aim at compromising intrusion detection systems. The proposed solution is to protect the integrity of the information shared among the CIDN peers, enhance their accountability, and secure their collaboration by thwarting insider attacks. A consensus protocol is proposed for CIDNs as a combination of proof-of-stake and proof-of-work protocols that enable collaborative IDS nodes to maintain a reliable and tampered-resistant trust chain. Their work focused on the theoretical aspects of security, to study a series of attacks reported in both domains (trust management and Blockchain), to fully understand the impact of various parameter choices on the proposed solution's security and the dynamics governing the trust score evolution. Although the research work has a prospect of securing shared information integrity from insider attacks, it failed to address external attack detection and data protection. Another piece in [50] presents a Blockchain-Based Malware Detection Framework (B2MDF) for detecting malicious mobile applications in mobile applications marketplaces (app stores). The framework consists of two internal and external private blockchains forming a dual private blockchain and a consortium blockchain for the final decision. The internal private blockchain stores feature blocks extracted by both static and dynamic feature extractors, while the external blockchain stores detection results as blocks for current versions of applications. The result showed that B2MDF does not limit the implementation of any specific machine learning algorithms.

B2MDF also provides useful features for third parties to develop their antimalware solutions as app stores may be more accurate in extracting features from a new uploaded sample.

In [25], The authors proposed a blockchain-based web attack detection model that uses a signature-based detection method. In their work, the signature-based detection looks for specific patterns against three known web-based attack types: Structured Query Language (SQL) Injection, Cross-Site Scripting (XSS), Command Injection. Three web servers are used for the experimental study. The installed blockchain node used multichain applications for each server. The attack signatures detected and defined by a web application are updated in the blockchain lists and used by all web applications. The result showed that the solution was able to detect these attacks. However, the approach failed to address signature sharing, and also, it cannot be applied to anomaly-based IDS. The authors in [51] proposed a novel business model of data sharing in multiple clouds, where a blockchain-based platform is designed to provide an ecological system. The model consisted of three types of participants: data owners, miners, and third parties. The data is shared via blockchain and recorded by a smart contract. These participants may acquire and store the sharing of data using their private or public clouds. They leverage the Shapley value to construct a dynamic and equitable incentive scheme for data sharing in multiple clouds. They analyze the participants' topological relationships and develop some Shapley value models from simple to complicated revenue distribution. The result showed that the reliable collaboration model provided innovations that realize the dynamic distribution of revenue by Shapley value. They proved that their solution could encourage more clouds to contribute their data and improved data authenticity as far as possible through verification and analysis. The method failed to address how participants are authenticated to deter or detect malicious activities.

In [52], the authors proposed the development of Distributed Intrusion Detection System (DIDS) using Blockchain over a stable platform like cloud infrastructure. In their work, they tested the performance of the overall system with Autoscaling. They showed the performance of the DIDS server with a varying load of data. They noted many other issues like communication delay, the overhead of blockchain, cost of

implementation, and so on, which must be discussed and analyzed. In [53], the authors investigated and demonstrated several collaborative intrusion detection methods to analyze blockchain's suitability and security for collaborative intrusion detection systems. They also studied the difference between the existing means of integrating intrusion detection systems with blockchain and categorized the significant vulnerabilities of blockchain with their potential losses and current enhancements for mitigation. Their survey recommended more studies in implementing CIDS with blockchain and other hardware security modules like HSM to elevate IoT devices' security, especially the newly emerging autonomous cars. The authors in [54] proposed the virtual reality parallel anti-DDoS chain design philosophy and distributed anti-D Chain detection framework based on hybrid ensemble learning. They utilized AdaBoost and Random Forest as their ensemble learning strategy, and lightweight classifiers such as CART and ID3 are integrated into the same ensemble learning algorithm. The blockchain scene's detection framework has much stronger generalization performance, universality, and complementarity to accurately identify the onslaught features for DDoS attacks in a P2P network. The experimental results confirmed that the distributed heterogeneous anti-D chain detection method performs better in six important indicators (such as Precision, Recall, F-Score, True Positive Rate, False Positive Rate, and ROC curve).

The authors in [40], [41], and [42] proposed the use of blockchain technology in detecting anomalies in computer networks. In [40], the authors proposed a blockchain anomaly detection (BAD) solution that focused on detecting attacks directed at the blockchain network. BAD prevented the insertion of a malicious transaction from spreading further in the Blockchain. BAD leveraged blockchain metadata named forks to collect potentially malicious activities in the blockchain network. Their works used machine learning to train blockchain nodes to detect malicious activities. In their approach, they considered an eclipse attack (an attacker infects a node's list of IP addresses, thus forcing the victim's node list of IP addresses to be controlled by an attacker). The result analysis showed that BAD could detect and stop the spread of attack that uses bitcoin forks to spread malicious codes. However, the solution is specific to attacks directed towards the blockchain network and that uses bitcoin forks. In another research put forward in [41], the authors proposed collaborative IoT anomaly detection via blockchain solution

(CIoTA). CIoTA uses the blockchain concept to perform distributed and collaborative anomaly detection on IoT devices. They used CIoTA to continuously train anomaly detection models separately and then combined their wisdom to differentiate between rare benign events and malicious activities. The evaluation of the result showed that combined models could detect malware activities easily with zero false positives. The proposed solution relies on IoT devices' collaborative effort to detect attacks; hence, it cannot share attack signatures or features among different nodes, and it is specific to malware attacks.

The authors in [42] proposed a blockchain-based malware detection solution in mobile devices. Their work extracted installation package, permission package, call graph package features for all known malware families for Android-based mobile devices and used them to build a feature database. Their result showed that the solution could detect and classify known malware. It can also perform malice determination and malware family classification on unknown software with higher accuracy and lower time cost. However, the solution is specific to signature-based malware attacks on Android-based mobile devices: hence, it will be challenging to apply it to anomaly-based attacks.

3.3 Blockchain Application in Healthcare

Healthcare industries generate a massive amount of data in the form of patient information daily, and this information needs to be secured, more specifically now that healthcare providers are proposing robust inter-healthcare operability. Due to data security reasons, blockchain becomes a tamperproof means of ensuring healthcare data integrity and consistency. Blockchain application in healthcare industries is still in its inception. However, the potential it offers, the deficiencies and gaps it fills (e.g., ensuring the security and confidentiality EHRs) place it at the forefront to be adopted by healthcare providers recently. Blockchain technology has been proposed in different kinds of researches to secure personal health records. Several kinds of research have proposed blockchain technology in the healthcare industry for data sharing. For instance, The authors in [55] propose a platform that enables a secure and private health record system by separating sensitive and non-sensitive data. The platform serves to share patients' healthcare data with researchers without revealing the patients' privacy. The model successfully uses proxy re-encryption

techniques to share a patient's sensitive data without revealing the private key and adopting an asymmetric cryptography technique to encrypt these data while storing it on the cloud. In a similar work in [56], the authors proposed i-Blockchain, which uses a permissioned blockchain to preserve the patient's health data (PHD) privacy and improve an individual's data exchange experience. It allows only qualified individuals and healthcare service providers (HSP) to join the network to prevent malicious intruders. It used cold storage functions as off-blockchain storage and hot storage functions as the store where users temporarily put requested data, a private key, and a public key for secure data exchange.

Furthermore, the authors in [57] proposed a conceptual design for sharing personal continuous dynamic health data using blockchain technology. The authors supplemented the approach with cloud storage. They used hash pointers to the storage location to solve the problem of sharing large-sized continuous-dynamic data while integrating blockchain and cloud storage. Extensive size data can be stored in an encrypted format on the cloud, and only the transactional data and metadata can be saved and shared on the blockchain. The authors in [58] proposed a decentralized record management system (MedRec) to manage authentication, confidentiality, accountability, and data sharing of EHRs using blockchain technology. It is a modular design that integrates with patients' local data storage and encourages medical stakeholders to participate as miners. The result showed that the system enables the emergence of big data to empower researchers while engaging the patient and providers in the choice of release metadata. A new approach which joins blockchain and cloud computing network was proposed in [59]. In their work, they employed Amazon Web Services and Ethereum blockchain to facilitate the sematic level interoperability of EHRs systems without standardized data forms and formatting. The model proposed an interoperability data sharing framework that includes security through multilayer encryption, optical data storage through Amazon Web Service, and transfer using the Ethereum blockchain. The authors in [60] proposed a blockchain platform architecture for clinical trial and precision medicine. The work identified four new architecture components required to build on top of traditional blockchain and discussed architectural challenges. The four components discussed in the paper are: (a) a new blockchain-based general distributed and parallel computing paradigm component to devise and study parallel computing

methodology for big data analytics, (b) a blockchain application data management component for data integrity, big data integration, and integrating disparity of medical-related data, (c) verifiable anonymous identity management component for identity privacy for both personal and IoT devices and secure data access to make possible of the patient-centric medicine, and (d) trust data-sharing management component to enable a trust medical data ecosystem for collaborative research.

In [61], the work described evaluation metrics to assess blockchain-based distributed applications (DApps) based on their feasibility, intended capability, and compliance in the healthcare domain. In their work, they highlighted seven crucial metrics for evaluating DApps designed to improve healthcare interoperability. The metrics highlighted in their work are (a) the entire workflow should be HIPAA compliant, (b) The blockchain platform should support Turing completeness, (c) DApp should support user identifiability and authentication, (d) DApp should support structural interoperability at minimum, (e) scalability across large populations of healthcare participants, (f) support of patient-centered care model, (g) cost-effectiveness compared to existing approaches. In [62], the authors presented a blockchain-supported architectural framework for secure personal data control in a health information exchange by pairing user-generated acceptable use policies with smart contracts. Their system adopted a user-centric approach for processing patient medical health data using processing nodes in the blockchain-based system. The scheme monitors data after reception by the requester and by policy destroys the data if violations occur or after the period allocated for computation expires based on smart contracts. They performed a comparative performance analysis of access control and privacy preservation considerations between their framework and other works from the literature. The result showed that their framework has a better performance than other works from the literature.

Despite the number of proposed solutions, blockchain applications have not addressed the security problems associated with data exchange in cooperative intrusion detection systems. Since the shared information is dynamic, needs to be accurate, tamperproof, and consistent, there is a need to continuously monitor the information for any malicious activities from both insider and outsider threat actors. This serves

as the motivation for this work. We propose an architecture that uses blockchain technology to secure and incrementally update attack features and signatures and detect any malicious activities in the distributed database in real-time. The proposed system facilitates scalable attack features or signature exchange, presents the information in a standard format that encourages heterogeneous IDS node participation, authenticates and verifies every transaction. It is also robust to public IDS nodes joining and leaving the blockchain network in real-time. The proposed architecture can ensure secure patients' health information exchange across different healthcare systems in real-time.

CHAPTER FOUR

METHODOLOGY

The proposed architecture, which is compatible with any blockchain platform, is built on the Ethereum blockchain platform. Ethereum blockchain is an open-source blockchain platform that can handle many concurrent transactions, making it scalable [63]. Apart from this, the Ethereum blockchain platform is flexible and compatible with standard operating systems. It is a blockchain-based distributed computing featuring smart contracts. A smart contract is an agreement among members of a consortium, stored on the chain and run by all participants [64]. Although the Ethereum platform is a public blockchain, in this work, we configure and run it as a private blockchain network that can accept public nodes without permission. Thus, the architecture possesses the features of private and public blockchain networks. We configured the Ethereum blockchain as a private blockchain by editing the configuration file that houses the genesis block. The customed configuration file makes our blockchain network invisible to random public nodes, and any node that wants to connect to the blockchain network will run the same customed configuration file. The customed genesis block ensures that other random nodes cannot interact with the blockchain network. It combines private, and public blockchain attributes to store and distribute cyberattack features and signatures securely. It is a private blockchain because only authorized nodes can prepare, verify, and validate transactions, while it is regarded as a public blockchain because public nodes do not need permission to join or leave the network. Figure 4 shows a pictorial representation of the proposed architecture.

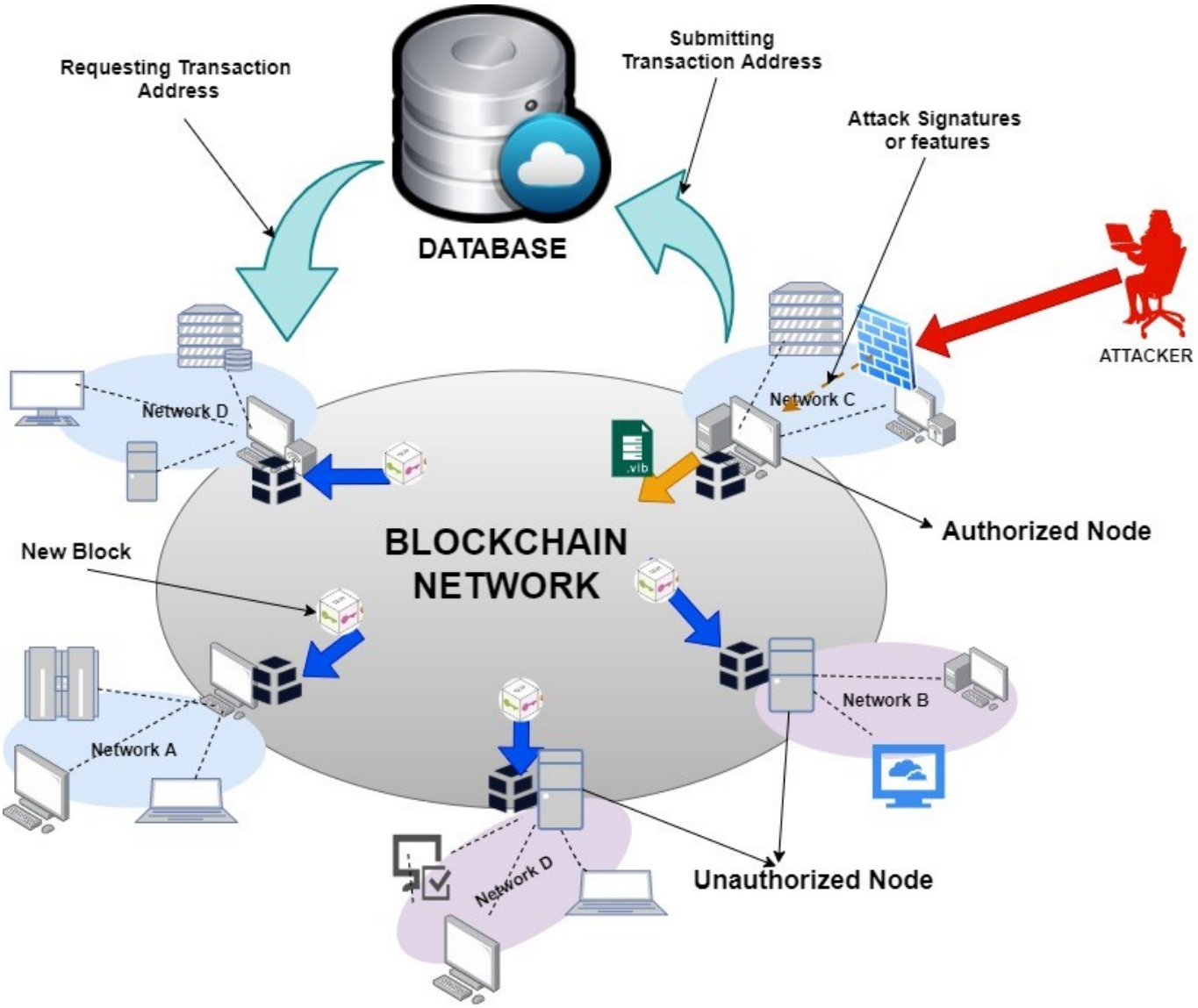

Fig. 4. The Pictorial representation of the Proposed Architecture

The architecture is composed mainly of the following:

- **Authorized Nodes**

  The authorized nodes are also known as miners. The nodes set up the terms and conditions in the smart contract. They are the trusted nodes that are privileged to prepare, submit, and verify transactions. They can be stand-alone nodes or gateways to other networks. They also run the consensus algorithm, thus validate transaction blocks. All authorized nodes update the database.

- **Unauthorized Nodes**

  These are also known as public nodes. They join the network to retrieve stored attack information. They can also be stand-alone nodes or gateways to other networks. Public nodes do not submit

transactions to the blockchain because they are not trusted. Hence, they are not privileged to prepare, verify, validate, or run consensus algorithm. They do not update the database but can only request the transaction address of the mined blocks.

- **Database**

  The database, which is accessible to all nodes, stores the address of the mined blocks. While all public nodes have read-only access to it, authorized nodes update block information. The database also stores the blockchain network's genesis block. The genesis block is required by every node that wants to join the blockchain network. Apart from the genesis block, the database houses information about published blocks. Any data manipulation in the database results in an inability to join or access the blockchain's contents but does not affect the stored data. Such malicious activity can be easily detected.

The proposed architecture is divided into three main stages, as shown below.

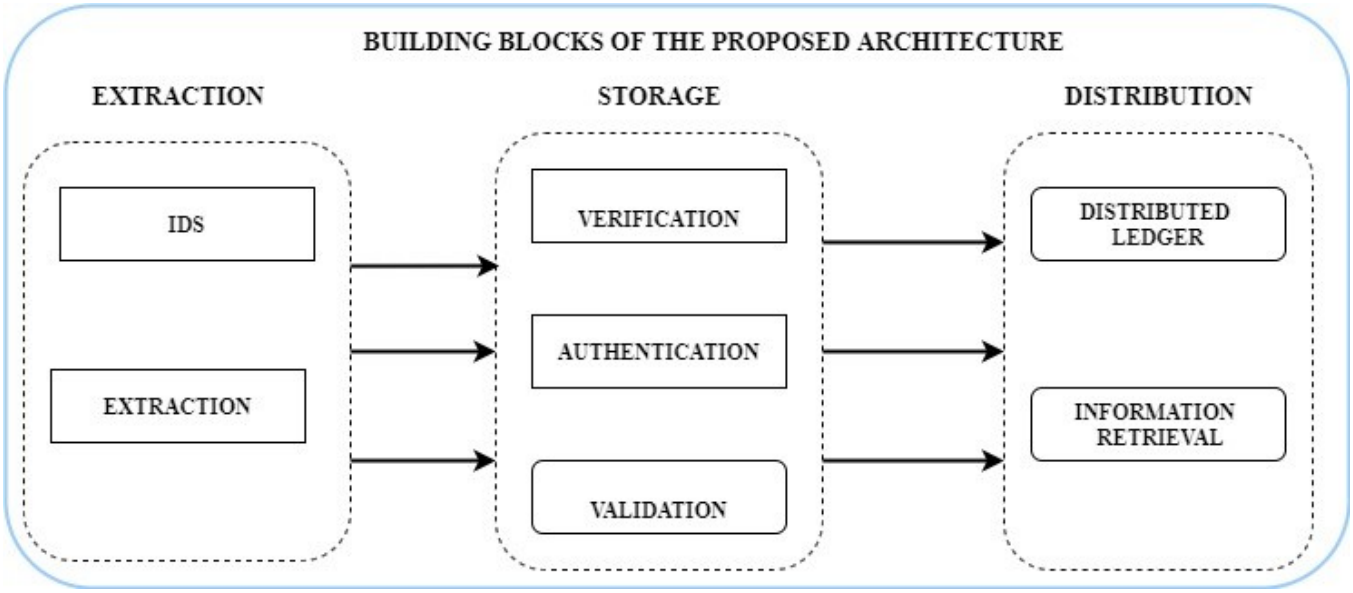

Fig. 5. Building blocks of the proposed architecture

4.1 Extraction

The extraction stage explains how signatures are retrieved from signature-based IDSs (details in Chapter Five). Attack signatures are rules written in a signature-based IDS to detect specific attacks. If the rules are correctly written, signature-based IDSs have the best accuracy in detecting known

attacks, but it is almost impossible to detect unknown attacks (zero-day attacks). The extraction stage explains attack launch on a host, type of IDSs utilized, the composition of attack rule, attack detection, and transaction formation. Furthermore, it explains features extraction from the anomaly-based IDS using developed scripts and network sniffing tools (details in Chapter six). Finally, we describe how the system prepares a transaction from the extracted signature or features and submits it to the blockchain platform for verification.

4.2 Storage

We describe the steps involved in verifying a submitted transaction and the transaction owner's authentication procedures. The section explains how the architecture authenticates every transaction sender and continuously monitors the node's behavior to detect malicious activities without a centralized authentication server. Instead of monitoring the node's behavior using a centralized server, the architecture achieves the result using a decentralized smart contract code running on the blockchain network. Furthermore, the section describes how the system converts successfully verified transactions to agreed-upon standard format using some novel approach. Finally, it explains the steps involved in transaction validation. The storage section explains how the system validates transactions using an iterative process such as a proof-of-work algorithm and incorporates a proof-of-stake algorithm to limit a miner's influence on a submitted transaction. We describe how the nodes confirm a block of transactions using nonce value and attach it to the chain.

4.3 Distribution

The distribution stage describes the steps involved in retrieving the content of a new block. The section describes how authorized nodes update the database and how transaction information is retrieved to query the blockchain network. Also, the distribution stage describes the information needed for a public node to join the blockchain network.

CHAPTER FIVE

ATTACK SIGNATURE ARCHITECTURE

Figure 6 shows the detailed processing blocks in the extraction, storage, and distribution of the attack signatures extracted from any signature-based IDS

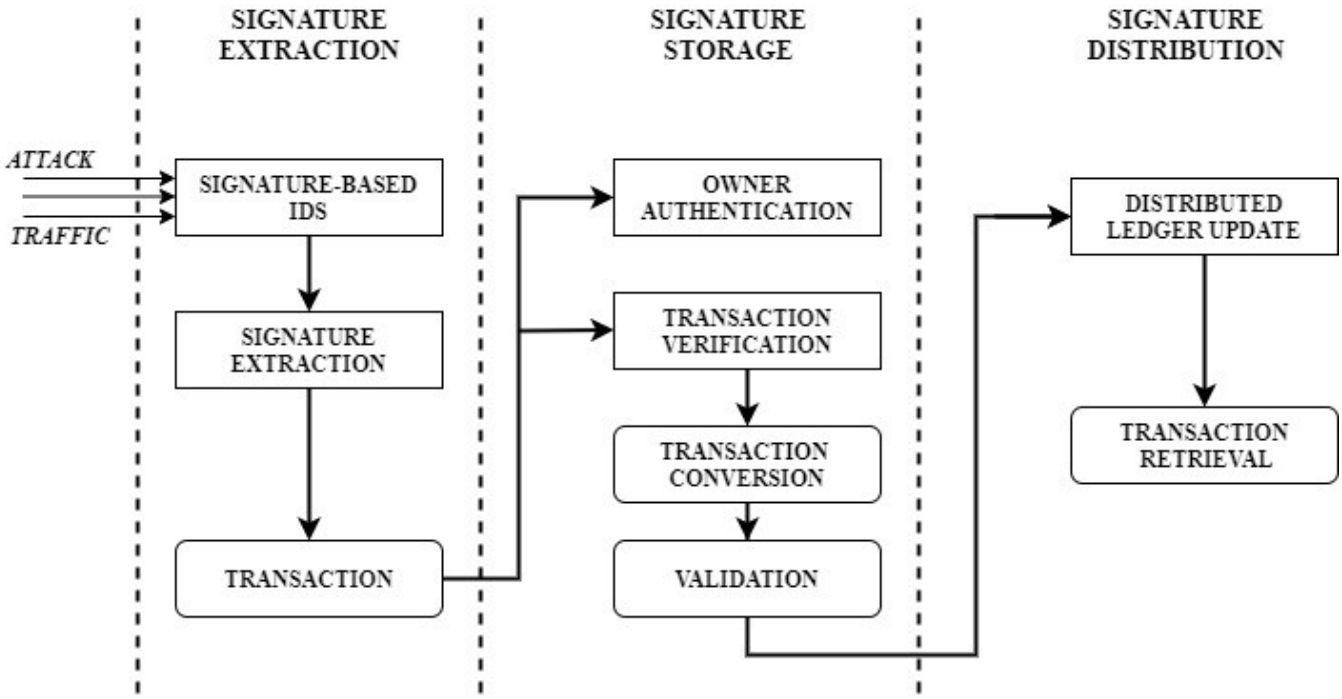

Fig. 6. Processing blocks of Attack Signatures

5.1 Signature Extraction

Attack signatures are rules written in signature-based IDSs to detect attacks. The IDS monitors the incoming network traffic and compares it to these rules. If the traffic pattern is similar to any of the predefined rules, an alert is generated. The attack signature that detects such an attack is retrieved from the IDS. In this work, we utilized three common signature-based IDSs; Snort[23], Bro[65], Suricata[66]. These IDSs are open-source IDSs that are downloaded and installed on our blockchain nodes. Rules to detect different types of attacks, such as Denial of Service (DoS), ICMP ping, etc., are written in the Snort IDS. We developed a python script that can extract these rules. Whenever an attack is detected, the specific rule used to detect such an attack is retrieved from the IDS using a developed script. Table I shows an example of a retrieved signature or rule.

Table I: Retrieved signature

S/N	IDS	SIGNATURE
1	Snort	*alert TCP ! $ any any -> $HOME_NET 80 (flags: S; msg:"Possible DoS"; count 70, seconds 10; sid:10001;rev:1;).*

- *S/N*: Serial number of retrieved signatures.
- *IDS*: The IDS that detects the attack. We experimented with Snort, Bro and Suricata.
- *SIGNATURE*: The IDS signature or rule that detects the attack.

A transaction is prepared based on retrieved signatures in Table I and signed by the owner using its private key. Every authorized node has unique key pair (public *pk* and secret *pk*), and each node is identified by its public key, transaction account, MAC, and IP addresses. The owner submits the digitally signed transaction to the blockchain network for verification. We introduce an extra step that authenticates the transaction senders to ensure that only authorized node transactions can be validated. As part of this action, the transaction owner submits a transaction tag (owner's details) that includes transaction account, MAC, and IP addresses alongside with transaction. Table II shows the transaction submitted to the blockchain network.

Table II. Format of Submitted Signature

S/N	Type	$SIGNATURE_{secret\_pk}$	Transaction Tag
1	**Snort**	*alert TCP ! $ any any -> $HOME_NET 80 (flags: S; msg:"Possible DoS"; count 70, seconds 10; sid:10001;rev:1;).*	**IP:** 192. 168.1.191 **MAC:** *00:1A:C2:7B:00:47* **Trans_acc:** *0x 8b695D0D7160aA8d95dc6ccEf6E7133F76a91De7*

5.2 Signature Storage

The transaction owner is authenticated, and the submitted transaction (i.e., Table II) is verified. The successfully verified signature is converted to an agreed-upon standard format and validated. All verification and signature conversion are handled on the server-side of the architecture, i.e., smart contract, while transaction validation is handled by blockchain consensus protocol. The accepted format of

submitted signature (Table II), public keys, and authorized nodes information are already coded in the smart contract. Also, signature conversion script (Algorithm 2) and signature format creation script (Algorithm3) are coded into the smart contract and mined to the blockchain. Thus, smart contract handles the following functions:

5.2.1 Node Authentication

The smart contract handles the node authentication. The purpose of authenticating the node is to ensure that all transaction submissions are restricted to authorized nodes only (i.e., it ensures that public nodes' transactions are detected). The smart contract retrieves the transaction tag and invokes the smart contract code that compares it with stored information. The information confirmed at this stage includes the transaction account, MAC and IP addresses, and the private key. We recognized that hackers could easily spoof the MAC and IP addresses in a highly unsecured environment; hence, we add a multifactor authentication process that involves verifying the transaction account and private key in addition to the verified MAC and IP addresses. The pseudocode in Algorithm 1 describes the snippet of the smart contract that handles the node's authentication. The smart contract invokes the code that reads the transaction tag. It verifies the digital signature using the owner's public key (*o.pk*) and confirms the transaction account, MAC, and IP addresses. If the node authentication is successful, the algorithm invokes the transaction verification code. For a node to be successfully authenticated, the sender must be authorized, and the digital signature must be verified. If any of these steps fail, the node's transaction is dropped.

Algorithm1: Node Authentication

```
Require:
    Mapping (pubkey=>bytes32) public Keys;
    Mapping (account => bytes32) public Account;
    Mapping (ip_addr=>bytes32) public IP;
    Mapping (mac_addr => bytes32) public MAC;

    procedure: ReadOwnerData (Transaction tag)
        Transaction.owner ←msg.sender
        Transaction.account ← account
        Transaction.pubkey ← o.pk
        Transaction.ip ← ip_addr
        Transaction.mac ←mac_addr
        Return TRUE
    end procedure

    procedure: NodeAuthentication (OwnerData)
        require (ip_addr ∈ IP)
        require (mac_addr ∈ MAC)
        require (pubkey ∈ Keys)
        require (account ∈ Account)
        require (o.pk verifies digital signature)
        return successful authentication with timestamp
    end procedure
```

5.2.2 Transaction verification and standard format creation

The transaction verification step ensures that all malicious transactions or activities on the submitted transaction by insider threat actors are detected and prevented. (i.e., it detects and prevents compromised authorized nodes from participating). This step also ensures that any submitted transaction's integrity and consistency are verified before attaching it to the blockchain network. We defined a code that checks the submitted signature's mandatory variables and converts it to the smart contract's agreed standard format. The piece of pseudocode in Algorithm 2 describes how a retrieved signature is converted to the standard format. The standard format is created based on what we defined as mandatory variables. Mandatory variables are variables that must be present in an attack rule. Examples of mandatory variables are action, source address, source port, destination address, destination port, message, protocol, flag, time, data rate, rev, and sid. Mandatory variables are needed for a signature to be able to detect an attack. These variables

give unique details about the attack and are similar in all types of signature-based IDS. The format creation script's snippet is developed based on the mandatory variables. The algorithm checks for these variables in the retrieved signature. If any of the mandatory variables are empty, the algorithm returns an error, and the signature is dropped. Otherwise, the algorithm reads these values and assign them to corresponding standard format variables. We ignore any other standard format variables without equivalent values from the retrieved signature, which might occur due to the different ways of writing attack rules.

Algorithm 2: Signature Conversion

Require:

```
Mapping(transaction => struct) public Signature;

procedure: ReadSigMandatoryVar (submitted signature)
        Signature.action ←action
        Signature.protocol ← protocol
        Signature.sourceip ←src_ip
        Signature.destinationip ← dest_ip
        Signature.sourceport ←src_port
        Signature.destinationport ← dest_port
        Signature.message ←msg
        Signature.sid ← sid
        Signature.rev ←rev
        return TRUE
end procedure

procedure: SignatureConversion (signature)
        require (action != NULL)
        require (protocol != NULL)
        require (src_ip != NULL)
        require (dest_ip != NULL)
        require (src_port != NULL)
        require (dest_port != NULL)
        require (msg != NULL)
        require (sid != NULL)
        require(rev != NULL
        Signature.push[values] => Format.values
        return Formatted signature
end procedure
```

5.2.3 Signature Validation

Blockchain protocols handle the validation of transactions. In this work, the blockchain platform uses both Proof-of-Work (PoW) and Proof-of-Stake (PoS). The pending transaction is built into a block, and the block is broadcasted into the blockchain network for validation. The authorized nodes (miners) work to validate the block. Figure 7 shows the process of building transactions into blocks and how blocks are chained together. Each block contains a unique code called hash; it also includes a hash of the previous block. Data of earlier blocks are encrypted or hashed into a series of numbers and letters. The authorized nodes work to get the target hash to validate a block. A target hash is a number that a hashed block header must be less than or equal to for a new block to be awarded. The miners achieve this target hash by using an iterative process such as PoW, which requires consensus from all authorized nodes. The PoW is a mathematic calculation performed by every authorized node that competes to validate a block. The characteristics of PoW are computationally difficult to compute and easy to verify. We set an upper bound of stake for every transaction to ensure fair competition among miners (i.e., to discourage authorized nodes with a larger stake from always emerge as the miner).

The process of guessing the hash starts in the block header. It contains a block version number, a timestamp, the hash used in the previous block, the hash of the Merkle Root, the nonce, and the target hash. Successfully mining a block requires an authorized node to be the first to guess the nonce and broadcast it to other nodes. Other trusted nodes verify the nonce value's correctness by appending this number to the block's hashed contents and then rehashing it. If the new hash meets the target's requirements, then the block is added to the Blockchain. The transaction is permanently stored on the blockchain network, and it is impossible to mutate/erase the block.

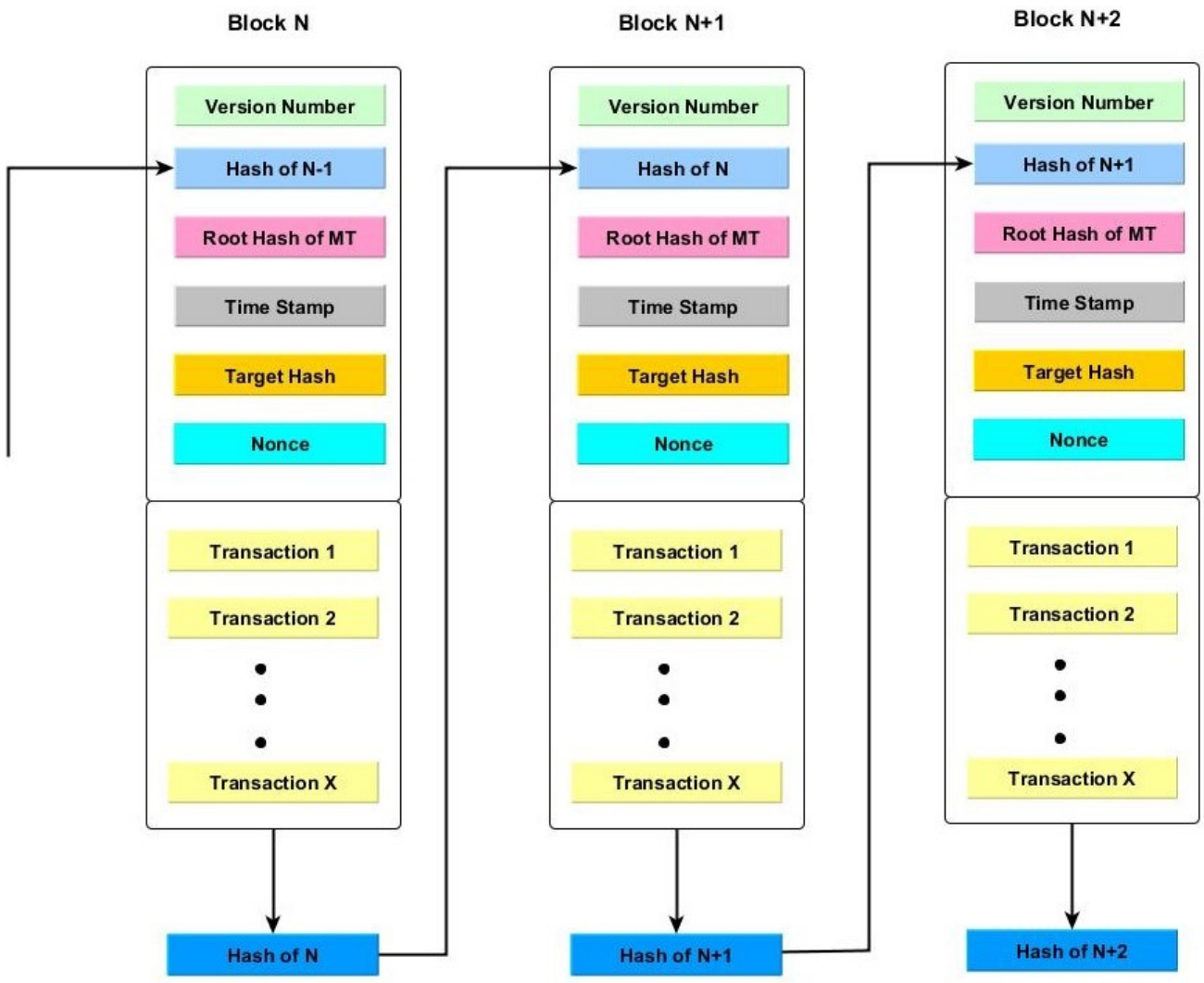

Fig. 7 Transaction Validation block

5.3 Signature Distribution

After a successful validation process, the blockchain issues a transaction address to the owner (sender). Each node's ledger is updated, and the transaction is ready to be retrieved. The steps involved in the secure distribution of mined signature are as follows:

5.3.1 Ledger Updating

The newly added block reflects on the ledger, which is possessed by every node in the network. The transaction address is sent to the database by the transaction owner. This database is opened to the public to obtain a copy of the block address.

5.3.2 Signature Retrieval

All blockchain nodes receive the newly added block notification but do not have access to the block's content. The transaction address obtained from the database is used to retrieve information stored in the new block. Nodes extract the stored attack signatures and convert them into the format of their signature-based intrusion detection systems.

5.4 RESULT

We implemented the proposed system on the Ethereum blockchain platform. We use Solidity *v 0.5.4* implementation of Ethereum for the smart contract and *geth v 1.4.18* for Ethereum. The Blockchain network was set up in the laboratory with five blockchain nodes, one database node, and one attack node. We implement four authorized nodes (to ensure consensus of miners during the validation stage) and one unauthorized (public) node in the setup. To make an authorized node, we write the node's information into the smart contract and mine it to the blockchain network. Before a public node joins the network, it retrieves a copy of the genesis block information from the database. Four nodes run Ubuntu Linux18.04, while one node runs Ubuntu Linux 16.04. Table III shows the configuration of each blockchain node.

Table III: Blockchain node configurations

Node	**OS**	**RAM**	**Processor**
Authorized node 1	Desktop, Ubuntu 16.04	4GB	2.2GHz
Authorized node 2	Laptop, Ubuntu 18.04	16GB	2.81GHz
Authorized node 3	Desktop, Ubuntu 18.04	8GB	2.44GHz
Authorized node 4	Laptop, Ubuntu 18.04	4GB	2.44GHz
Unauthorized node	Laptop, Ubuntu 18.04	4GB	2.40GHz
Attack node	Laptop, Ubuntu 16.04	4GB	2.20GHz
Database node	Desktop, Windows 10	4GB	i5 @2.44GHz

We randomly installed the three signature-based IDSs on the blockchain nodes. *Snort v2.9.7* was installed on authorized nodes 1, 2, and the unauthorized node. *Bro v2.6.1* was installed on authorized nodes 2, 3,

and unauthorized node, and *Suricata v 4.1.3* was installed on authorized node 4. The transaction accounts, public keys, IP, and MAC addresses of all authorized nodes are written in smart contract lists and mined into the blockchain. In authorized node 2, we set a rule to detect a denial of service (DoS) attack on the local rule file of its snort IDS, and we start Snort in monitoring mode.

Attack rule*: alert tcp ! $ any any -> $HOME_NET 80 (flags: S; msg:"Possible DoS"; flow: stateless; threshold: type both, track by_src, count 70, seconds 2; sid:10001;rev:1;)*

Attack node launches a DoS attack to authorized node2 using hping3 syn packet.

DoS attack: *hping3 -i --flood -S -d 120 -p 80 192.168.1.144(Home_net)*

The target (authorized node 2) detects the attack, retrieves the signature, and submits it as a transaction to the blockchain network. The transaction is verified, converted to a standard format, and distributed among other blockchain nodes. The retrieved standard format transaction is shown in Table IV. Based on the Table, we can conclude that the architecture can present attack signatures in a standard format that other IDS nodes can easily understand, thus, enhancing heterogeneous IDS participation.

Table IV: The formatted retrieved signature

Standard Format Variable	**Signature Values**
Action	Alert
Protocol	TCP
Source IP	Any
Source port	Any
Destination IP	Home_net
Destination port	80
Flags	S
Message	Possible DoS
flow	------
Data rate	>70
Time (seconds)	2

sid	10001
rev	1

5.4.1 Performance Analysis

We defined the transaction dissemination latency as the metric for measuring the performance of the architecture. To evaluate the transaction latency, we obtain the following data for each transaction from every node in the blockchain network.

- *Transaction deployment time (t_1):* This is the time a transaction is submitted to the network. These data are collected directly from the sender terminal.
- *Execution time (t2)*: This is the time taken for the formatted signature to appear in each node's designated files. The time is retrieved by setting on current time on all nodes.

5.4.1.1 Transaction Dissemination Latency

The transaction dissemination latency is also known as the response time (measured in seconds) of the blockchain network. For each transaction, latency is the difference between the execution time and the deployment time (t_2-t_1). Latency includes the time elapse for authentication, verification and conversion, mining, and time spent to retrieve mined signatures. Figure 8 shows the response time of each node for every transaction. We observed that the 6^{th} transaction latency was the smallest while the 4^{th} and 8^{th} transaction has the highest latencies for all nodes. It has been shown that mining time significantly influences blockchain network latency since it involves iteration-thus it is not deterministic for all transactions. Also, verification time (format conversion) influences the network latency. The high dissemination latency, therefore, could be a result of the high mining time. Figure 9 shows the average response time of each node for twenty consecutive transactions. The average response time is the addition of all response times for each transaction divided by the number of transactions. We can see that the average response time for all nodes is less than 3 seconds.

Furthermore, figure 10 shows the percentage of time elapse at each stage of the transaction processing. The graph shows the percentage of time expended in each stage. In the verification stage that involves authentication of the sender, transaction verification, and conversion, 39% of the time was spent. While in the validation stage, which involves an iterative process of guessing the target hash, 60% of the time was spent, and the update stage only expends 1% of the time.

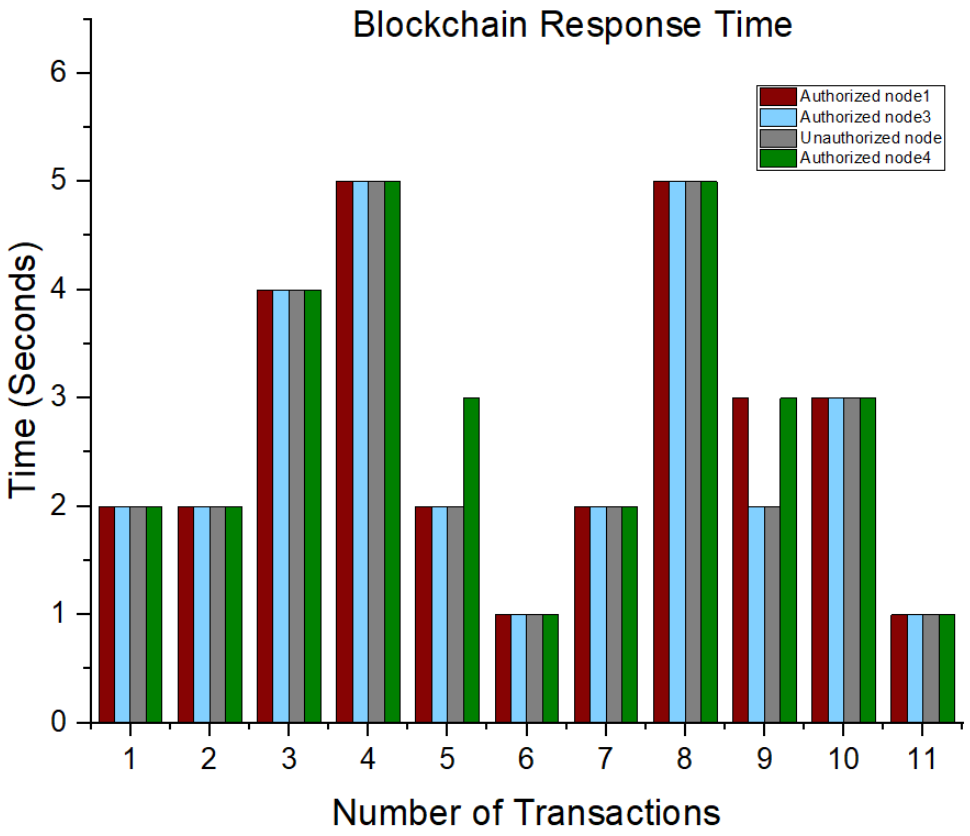

Fig 8. Dissemination latency for Transactions

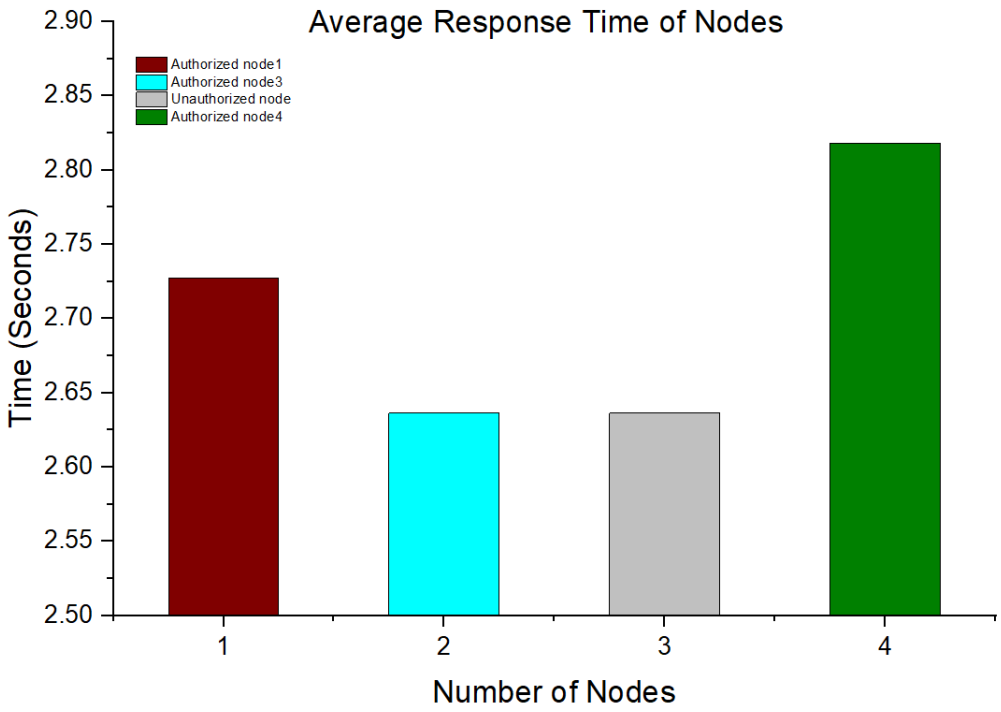

Fig 9. Average dissemination latency of each node for 20 transactions

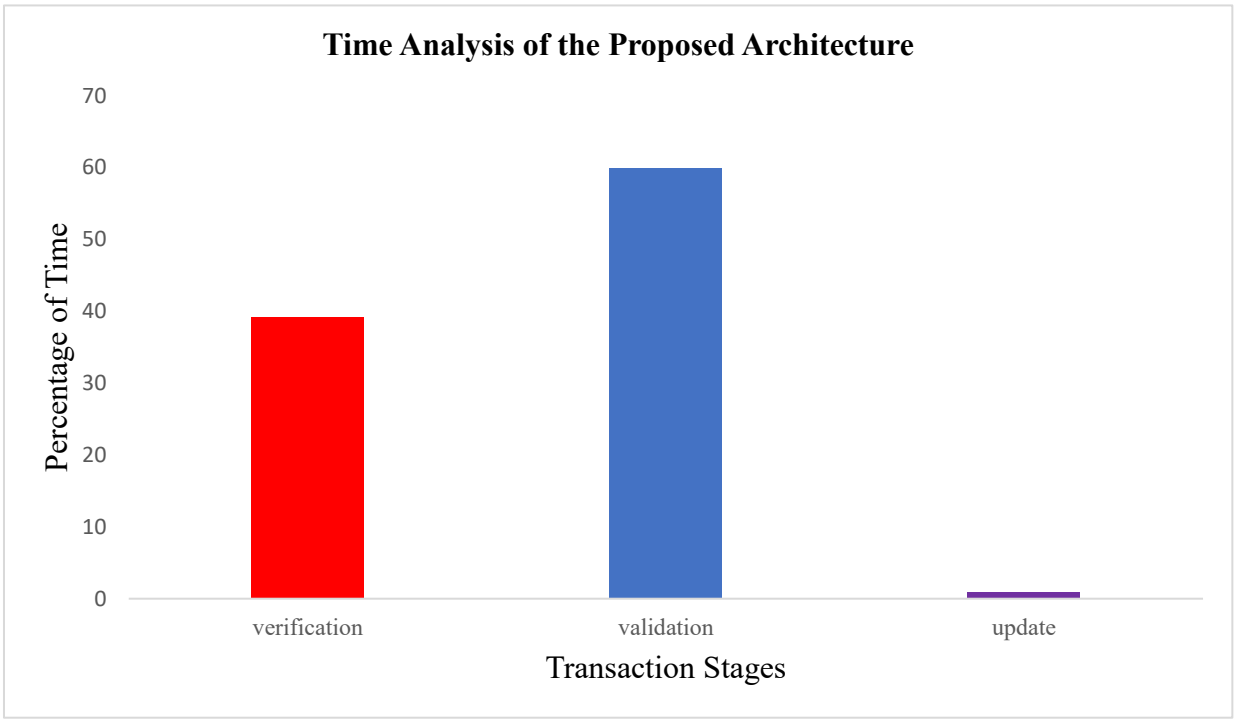

Fig 10. Time Analysis of Transaction stages

CHAPTER SIX

ATTACK FEATURES ARCHITECTURE

The features building block is divided into three main steps, as shown in Figure 11. The owner authentication, validation, and transaction distribution processes follow the same description as in Chapter five. This chapter focuses on feature extraction, verification, conversion processes, and the results.

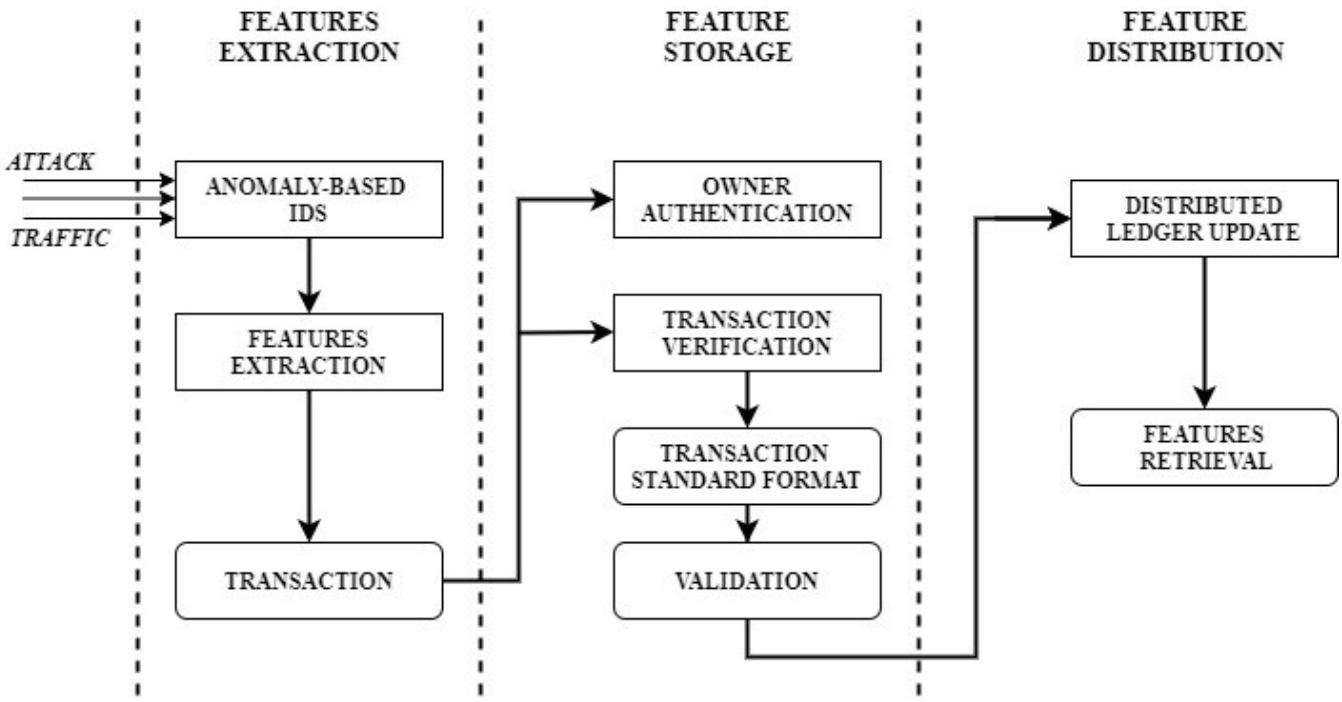

Fig. 11. The Building blocks of Features Distribution

6.1 Features Extraction

Network traffic is made up of connections and packets sent between the hosts. Every network traffic or packet has characteristics that distinguish it from other traffic; these characteristics are known as features. Attack features are distinguishing information of attack traffic that differentiates them from regular traffic. Anomaly-based IDSs are trained to identify attack traffic. They are trained with features extracted from regular (standard) traffic to establish a regular pattern and used the pattern to detect any known traffic pattern deviation. Anomaly-based IDS has been shown to have the capabilities of detecting zero-day attacks with high accuracy [9,10]. In this work, we analyze the network traffic when IDS alerts and extract

the features based on attack feature description in the KDD dataset [67] using network traffic analyzing tools. Attack features are extracted under two categories: (i) Connection features and (ii) packet features.

6.1.1 Connection Features

These features are obtained from connections an attacker makes with the network. We developed a script that analyzes attack connections in real-time. The script continuously sniffs the network traffic while waiting for an IDS alert. Whenever an attack is detected, the script captures and analyzes network connections using *tcpdump v4.9.2., libpcap v1.9.0, tcptrace v6.6.0,* and *Wireshark v3.0.1*. Tcpdump captures and analyzes TCP packets, while Wireshark uses libpcap to capture network connections in real-time. Tcptrace is used to analyze the captured connections. Some of the features extracted from attack connections are shown below in Table V.

Table V: Features generated from attack connections

S/N	Feature Name	Definition
1	Source Port	Port from which attack is launched.
2	Destination Port	Target port in the target network.
3	Source IP	The IP address of the attack node.
4	Destination IP	Target IP address in the target network
5	Source Bytes	The total number of bytes sent from attack nodes.
6	Destination Bytes	The total number of bytes sent from the target network to attack nodes.
7	Source Packets	The total number of packets sent from attack nodes.
8	Connection	The total number of connections initiated with the target network by attack node.
9	Duration	Total time elapsed during an attack.
10	Packets/seconds	The number of packets sent by an attack node within 1 second.
11	Source Host count	The total number of attack nodes connecting to the target network.
12	Destination Host Count	The total number of target nodes in the target network.
13	Throughput	The rate at which attack nodes send bytes to the target node. (measured in kbps).

14	Service Count	The total number of ports connected to by attack nodes during the attack period.
15	Same service count	The total number of connections to the same port number during the attack period.
16	Different Host rate	Percentage of attack nodes attacking different target nodes.
17	Same service rate	Percentage of attack nodes attacking the same port.
18	Same Host rate	Percentage of attack nodes attacking the same target node.

6.1.2 Packet Features

These features are obtained from the packets sent to a target node from the attack node. These features are obtained by capturing and analyzing attack packets' header. A snippet of the developed script listens for IDS alert and starts capturing and analyzing ingress packets using Scapy v 2.4.0 in real-time. Scapy decodes traffic packets and matches request with replies. Whenever an IDS alerts for an attack, the script starts capturing, analyzing, and extracting the attack packet headers' defined features. Table VI shows some of the packet features extracted.

Table VI: Features generated from attack packets

S/N	Feature Name	Definition
1	Land	'1' if the source and destination IP and ports are the same; otherwise, '0'.
2	Type of service	Class of traffic assigned to attack packet
3	Protocol	Higher layer protocol used in the data portion of the attack packet
4	Ip flags	How packet should be routed or processed by a higher layer
5	TCP Flags	Defines the type of packet sent by the attack node
6	Urgent	Indicates priority of handling packets by the router
7	Time to Live	Time left for a packet to be discarded
8	Checksum	Error checking in the packet header
9	Wrong Fragment	'1' if the checksum is 'incorrect'; otherwise, '0.'

Based on feature values in Tables V and VI, a node prepares a transaction, signs it with the private key, and submits it to the blockchain network for verification. As discussed earlier, the transaction sender submits a transaction tag that contains the information about the sender. Table VII presents the transaction submitted to the blockchain network for verification.

Table VII: Submitted Transaction

Class	**(FEATURES)priv_key**	**Transaction tag**
DCA	Signed Features (Combination of Tables V and VI)	**IP:** 192. 168.1.191 **MAC:** *00:1A:C2:7B:00:47* **Trans_acc:** *0x 8b695D0D7160aA8d95dc6ccEf6E7133F76a91De7*

- *Class*: This describes the type of anomaly-based IDS. In this case, it indicates Dendritic Cell Algorithm (DCA).
- ***(FEATURES)****priv_key*: This is the signed features. The signed transaction combines the packet and connection feature names
- *Verification information*: Additional security information

6.2 Storage

The conformity of the submitted transaction with feature format is first verified. In this case, the algorithm ensures that no features field is empty or missing. The architecture also verifies the sender's privilege to submit transactions and the gas price of mining the transaction. The smart contract converts successfully verified transaction to the agreed-upon standard format useful by other nodes and push it to the validation stage. (i.e., attached to the blockchain). The storage stage is divided into the following steps:

6.2.1 Transaction and Owners verification

The node authentication process is discussed in chapter five. The feature verification procedure ensures that all malicious transactions or activities on the submitted features are promptly identified and prevented. (i.e., it prevents malicious authorized nodes from compromising submitted features). The verification stage ensures that any submitted transaction conforms with the agreed-upon standard format, which other IDS nodes can use. The verification also ensures that there is enough information to identify such attacks by other IDS quickly. In the smart contract, we defined the threshold gas price of mining a transaction

since transactions are submitted in a similar format, making their prices almost identical. Any transaction costs higher than the threshold are flagged and results in a failed transaction. To ensure fair mining competitions among authorized nodes, we limit the stake that a node can put on a transaction, and we also define who should not mine a submitted transaction. Here, we set a mining policy that alerts when transaction owners attempt to mine their transactions. The smart contract review and compares the stake of each transaction and keeps track of all nodes participating in mining a transaction.

Furthermore, we defined the format for submitted transactions. The structure of the format contains the features name in Tables V and VI. A snippet from the smart contract runs through the submitted transaction to ensure no missing feature values are present. We generate the key pairs using Digital Signature Algorithm (DSA) (details in next chapter). The public key of all the authorized nodes is written in the smart contract, while the private key is kept securely within the nodes. A snippet from a smart contract shows algorithm 4 pseudocode that describes the verification process. For features verification to be successful, no feature fields must be empty (i.e., all feature fields must have values), and the transaction owner must not mine its transaction. Also, the stake and cost of mining any transactions must not exceed the threshold. If any of these conditions fail, the transaction fails, and it is dropped. A successfully verified transaction invokes a code that converts it to a standard format. The formatted transaction is then pushed for validation.

Algorithm 4: Transaction Verification

Require:

Mapping(data => struct) public Features;
Mapping (miners => struct) public Miners

procedure: ReadFeature (*Submitted transaction*)
Transaction.format ← feature_fields
Transaction.gasprice ← cost
Transaction.miner ← miners
Transaction.owner ← msg.sender
Miner.stake ← stake
return TRUE
end procedure

procedure: FeatureVerification (*Transaction)*
require (feature_fields != NULL)
require (cost <= ThresholdCost)
require (stake <= ThresholdStake)
require (msg.sender ∄ Miners)
Transaction.push[Features_values] => Standard Format
return Formatted Transaction with timestamp
end procedure

6.2.2 Features Validation and Distribution

The pending transaction is built into a block, and the block is broadcasted into the blockchain network for validation. Every node receives a broadcasted block, but only authorized nodes (miners) work to validate the block. The validation and distribution process follows the same steps as described earlier in chapter five.

6.3 RESULT

We implemented the proposed system in the laboratory and on the google cloud platform. The aim is to compare the architecture's behavior of the two deployments (i.e., when the nodes are close to each other and far apart). We set up eight blockchain nodes, one database node, and one attack node for the lab implementation while utilizing seven blockchain nodes in the google platform. We deployed one blockchain node in each available seven google cloud regions around the United States for the cloud implementation. Table VIII shows the configuration of lab nodes, while Figure 12 shows the cloud

deployment locations. We use Solidity *v 0.6.2* implementation for smart contracts and *geth v 1.9.0* for Ethereum.

Table VIII: Node configurations for laboratory Implementation

Name	**Machine**	**OS**	**RAM**	**Processor**
Node 1	Desktop	18.04	4GB	2.2GHz
Node 2	Laptop	18.04	16GB	2.81Ghz
Node 3	Desktop	16.04	8GB	2.44GHz
Node 4	Laptop	18.04	4GB	2.44GHz
Node 5	Vmware	18.04	4GB	2.2GHz
Node 6	Vmware	18.04	4GB	2.2GHz
Node 7	Vmware	18.04	4GB	2.2GHz
Node 8	Vmware	18.04	4GB	2.2GHz
attacker	Laptop	16.04	4GB	2.2GHz
database	desktop	window	4GB	I5@2.44

Fig. 12. Location of blockchain nodes around the United States

We run the features extraction scripts that use *tcpdump v. 4.9.2., libpcap v. 1.9.0, tcptrace v.6.6.0*, *Wireshark v. 3.0.1*, and *Scapy v.2.4.0* on all authorized nodes. For the proof of concept, we also run an anomaly-based IDS called Dendritic Cell Algorithm (DCA) [9] on one authorized node (node 2). The attacking node launches a DoS attack at node 2; the scripts extract the features and submits them to the blockchain network. The transaction is verified, validated, stored, and distributed among other nodes.

We perform other frequently encountered attacks such as port scanning and Land attacks on node 2 to extract the features. We record the transaction dissemination latency of each node for every transaction and calculate the average latency. Table IX shows the feature values for each attack. We implement moderate network traffic in the lab where every node has similar network traffic; however, the cloud experiment features diverse network traffic for each node. We evaluate the architecture's performance using dissemination latency and scalability metrics.

Table IX: Extracted Features for DoS, Port scanning and Land Attacks

S/N	**FEATURES**	**DoS**	**Port Scanning**	**Land**
1	No of connections	6594	8	11
2	Source bytes(kbytes)	1147008	708	846
3	Source frames	6592	10	9
4	Source throughput(kbps)	1698.4	216.1	917.96
5	Source frame/second	9995.4	3125.0	9999.99
6	Destination bytes(kbytes)	355968	364	0
7	Destination frames	6592	6	0
8	Destination throughput(kbps)	527.1	111.08	0
9	Destination frame/second	9995.4	1875.0	0
10	Duration(seconds)	0.65	0.0032	0.009
11	Source diff. host rate	0%	16%	11%
12	Source same host rate	100%	84%	89%
13	Source diff. service rate	99%	100%	11%

14	Source same service rate	1%	0	89%
15	Source diff. host count	1	1	1
16	Source count	6583	6	1
17	Destination diff. host count	1	1	1
18	Destination service count	1	3	1
19	Source IP	192.168.0.144	192.168.0.144	192.168.0.161
20	Source port	8131	48314	80
21	Destination IP	192.168.0.161	192.168.0.161	192.168.0.161
22	Destination port	21	22	80
23	Protocol	TCP	TCP	TCP
24	Type of service	0	0	0
25	Time to live	64	64	64
26	TCP flags	SYN	SYN	SYN
27	IP flags	RES	DF	RES
28	Urgent	0	0	0
29	Fragment	0	0	0
30	Land	0	0	1
31	Checksum	Correct	Correct	Correct
32	Wrong fragment	0	0	0

6.3.1 Performance Metrics

6.3.1.1 Response Time

We measure the response time of the architecture in two different scenarios: (i)Closed proximity, i.e., when we set up the experiment in the lab, and (ii) Wide geographical area, i.e., when deploying the experiment to the cloud platform. In the initial evaluation, we limit the cloud experiment to regions around the United States, thus deployed it to seven available regions. Figure 13 shows the average response time of all nodes for both laboratory and cloud deployments. We could observe that the average response time difference is not significant for both cases, contrary to expected. The reason for slightly low latency in the

cloud deployment might be that the cloud nodes are connected with high-bandwidth, low latency dedicated optic fibers that provide high throughput between them. In addition to this, the cloud nodes' computing power might contribute to a slight reduction in the latency as compared to the lab nodes. Based on the result, we can conclude that the architecture can facilitate scalable attack features or signature exchange among IDS nodes irrespective of the node's location.

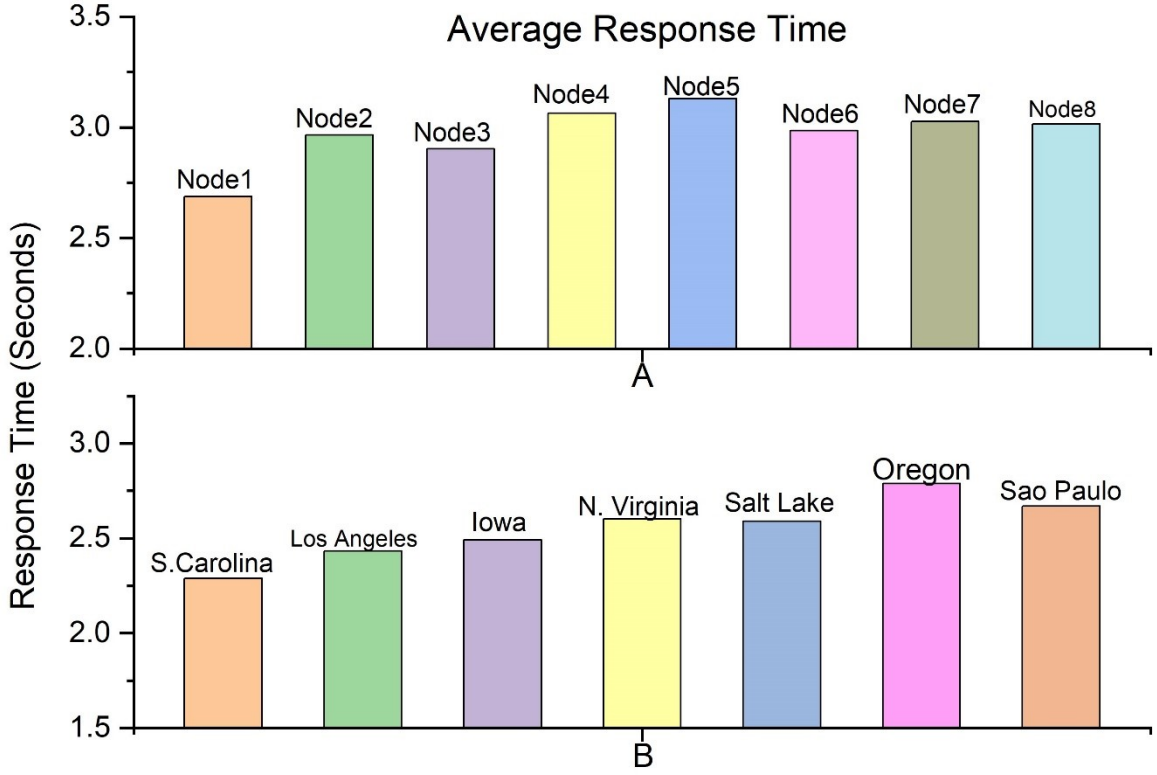

Fig. 13. A) Average response time when performed in the lab. B) Average response time when deployed to locations around the US.

6.3.1.2 Scalability

We evaluate the change in the latency of the architecture with an increasing number of nodes. We first implement increasing the number of unauthorized (public) nodes on authorized nodes' response time. We randomly choose two nodes that are in South Carolina and Los Angeles and make them authorized nodes. The blockchain network is step up as described above with these authorized nodes, and an attack node launches a DoS attack. The features extracted from DoS attacks are prepared as transactions and submitted to the blockchain network. These transactions are verified, validated, stored, and distributed among the

blockchain nodes. We record the deployment and execution times of each transaction for the two authorized nodes. We increase the number of public nodes joining the network one at a time, repeat the experiment, and evaluate the two authorized nodes' response times. Figure 14 shows the response time of the two authorized nodes for an increasing number of public nodes. We observed that increasing the number of public nodes has no significant effect on the architecture's response time, which implies that the solution is robust to public IDS nodes joining or leaving the network.

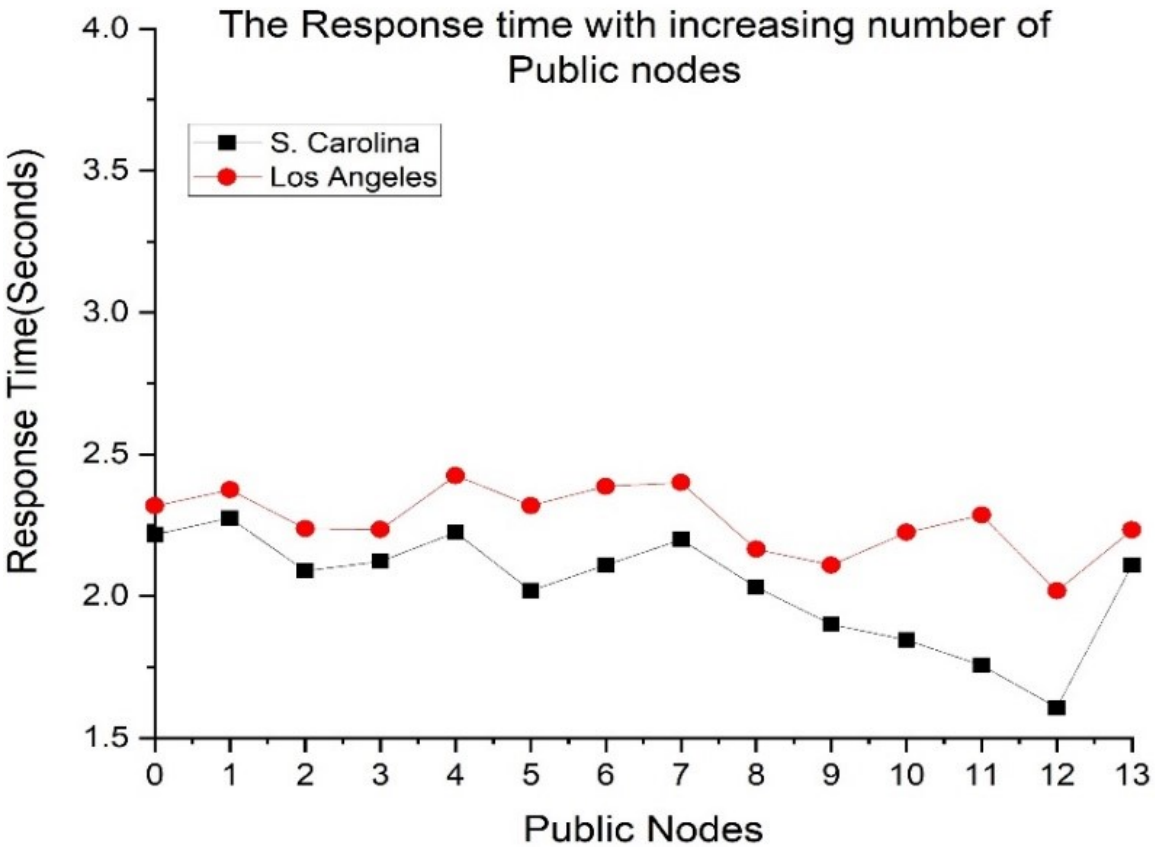

Fig.14. The response time with an increasing number of public nodes

Furthermore, we evaluate the response time with an increasing number of authorized nodes (i.e., miners). We set up a blockchain network in the lab, as we described above. We repeat the experiment and records the deployment and execution times of each transaction for all the nodes. We repeat the experiment several times, and the average response time of each node is recorded. We started with two miners and increased the number of miners one at a time. We repeat the experiment and record the average response times of each node. We repeated the experiment when we deployed it to the

cloud platform and compared the results. Figure 15 shows the response time of the nodes as the number of miners increases for both deployments. We observe a slight fall in the response time as more authorized nodes are added to the network. The decrease in latency might be due to more miners' availability to compete in the validation process, reducing the mining time (which accounts for an integral portion of the latency). The result further confirms that the architecture can facilitate scalable and prompt attack features exchange among IDS nodes.

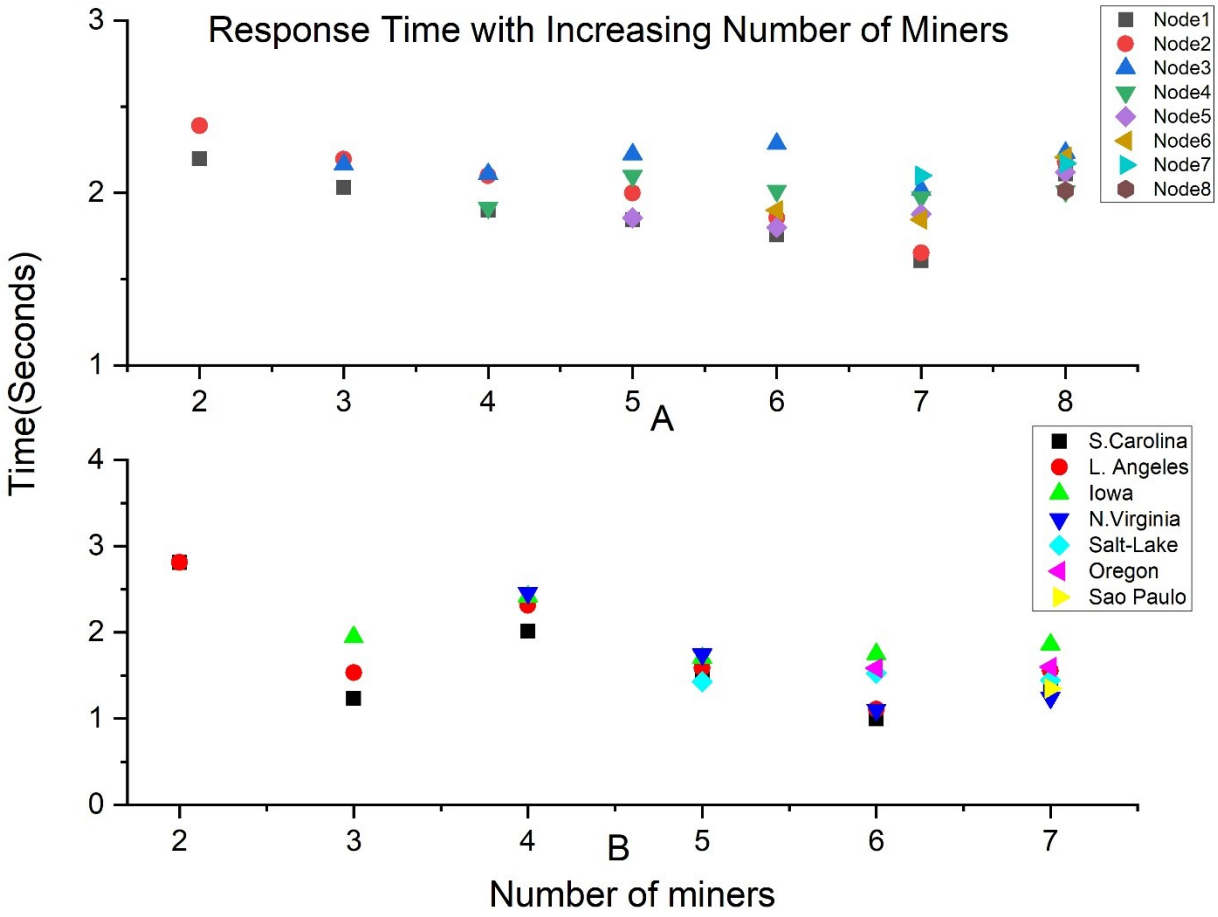

Fig. 15. A) Response time of nodes with an increasing number of authorized nodes for closed proximity implementation (B) Response time of nodes with an increasing number of authorized nodes for wide-area deployment

CHAPTER SEVEN

ARCHITECTURE'S SECURITY ANALYSIS

7.1 Security Analysis

The chapter describes the methods used in the proposed architecture to detect malicious activities from compromised authorized (insider threats) and untrusted public (external threat) nodes. In a private blockchain, the architecture is built to keep public nodes from interacting with the blockchain; hence they do not bother about detecting external threats. On the other hand, public blockchain incorporates every node; thus, it is challenging to isolate malicious nodes. Since the proposed architecture incorporate both private and public nodes, it is crucial to ensure that it can detect malicious activities from both internal and external threat actors. Figure 16 shows the authentication and verification processes to secure the network, especially from compromised authorized nodes. This chapter focuses on a detailed explanation of the transaction verification stage, as validation and distributed ledger stages have been described in the previous chapters. The verification is divided into node authentication and transaction verification.

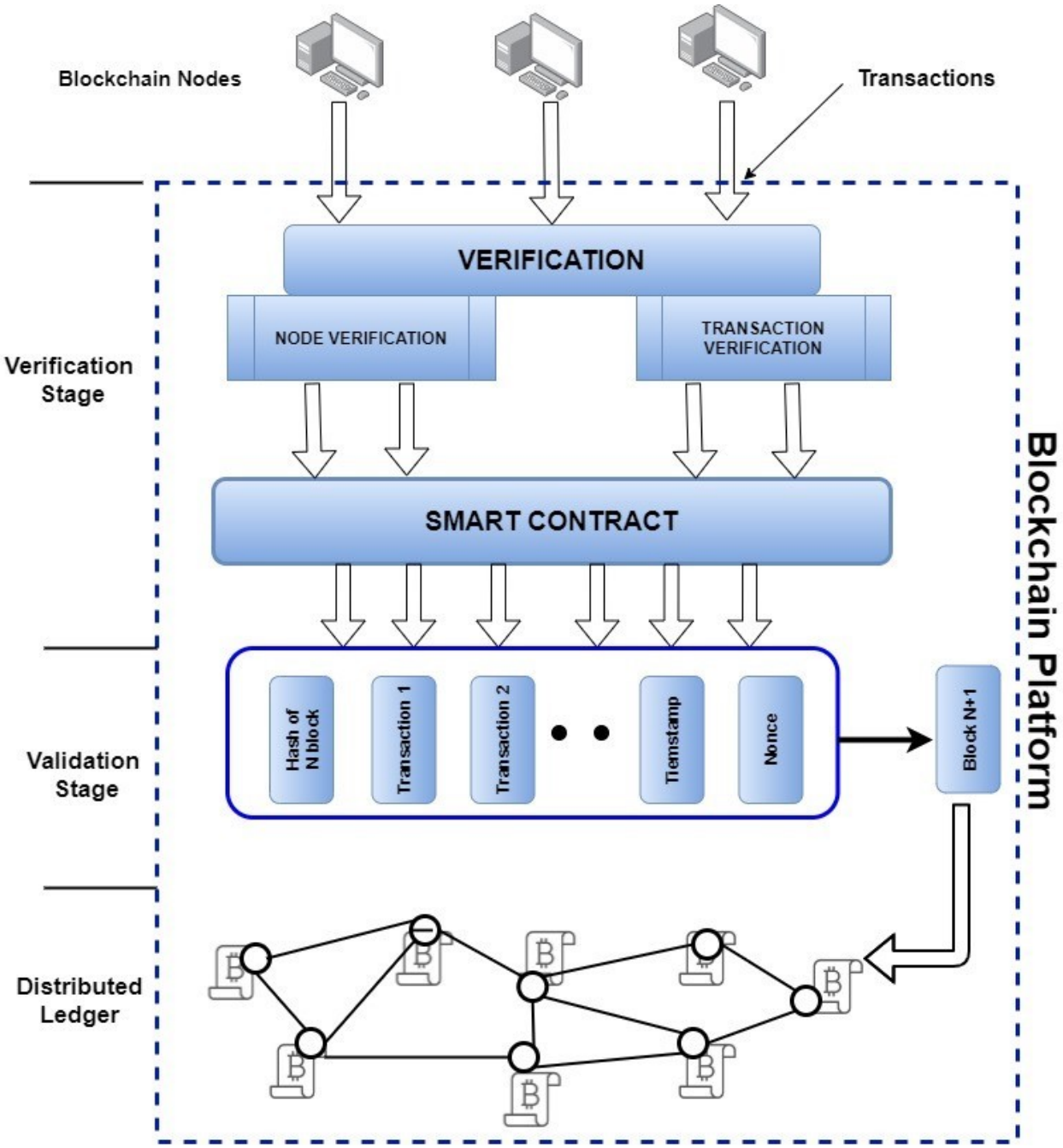

Fig. 16. Authentication and Verification blocks

7.1.1 Node Authentication

The smart contract authenticates every node that submits a transaction. In the blockchain network, nodes sign transactions using asymmetric cryptography: only the private key owner can create and sign her transactions. The person that signs a particular transaction can be verified using the corresponding public key. A public key acts as an address that belongs to the owner; it can be freely propagated in the blockchain network. However, the public key is just a bitstring and does not typically reveal the actual owner's true

identity. In this work, the public keys of all authorized nodes are listed in the smart contract, and when a node submits a signed transaction, the smart contract executes the code that verifies the sender's private key. We also implement a type of policy that evaluates transaction access control management in a consortium blockchain network. Authenticating a node requires that the smart contract retrieves the node information (transaction tag) and invokes a code that compares it with authorized nodes' listed information. The pseudocode below describes the snippet of the smart contract that handles the node's authentication. If a node is successfully authenticated, the algorithm invokes the transaction verification code; else, the transaction is dropped.

Algorithm 5: Node Authentication

```
Require:
        Mapping (pubkey=>bytes32) public Keys
        Mapping (account => bytes32) public Account;

        procedure: ReadTransactionTag (Data, account, pubkey)
                Data.owner ←msg.sender
                Data.account ← account
                Data.pubkey ← pubkey
                Return TRUE
        end procedure

        procedure: NodeAuthentication (pubkey, account)
                require (pubkey ∈ Keys)
                require (account ∈ Account)
                Owner.pubkey verifies owner.privkey
                return successful authentication with timestamped
        end procedure
```

7.1.2 Transaction Verification

The smart contract acts as a firewall that continuously monitors the ingress traffic (transaction submission) to the blockchain network. The smart contract handles the transaction verification before building it to a block. The transaction verification step ensures that all malicious transactions or activities on the submitted transaction, especially from insider nodes, are identified and thwarted. (i.e., it detects malicious activities

from a compromised authorized node). The verification step also ensures that any submitted transaction's integrity and consistency are verified before attaching it to the blockchain network. This characteristic is primarily applicable in the architecture's application in the Healthcare industry. To set up the smart contract, we defined a maximum gas limit for mining a transaction since transactions are submitted in a specific format; thus, gas prices almost identical. Any transaction gas prices higher than the threshold is flagged and results in failed transactions. To thwart the malicious behavior of transaction owners, we defined a function that keeps track of the miners. Here, we define a policy that invokes and drops transactions when transaction owners attempt to mine their transactions. We also set the maximum stake a miner can place on any transaction to encourage fair competition among the miners. The smart contract checks each submitted transaction's gas price and keeps track of all nodes participating in mining the transaction.

Furthermore, we defined the format for submitted signatures and features and set a rule limiting the maximum number of transactions per second from a node. For each authorized node, we generate the key pairs using Digital Signature Algorithm (DSA) with 1024 bit-length using the command *ssh-keygen -t dsa -b 1024*, copied the public key, and write it in the smart contract. Every authorized node signs its transaction with the private key, and a smart contract code verifies it using their public keys when the transaction is submitted to the blockchain. We also defined a retrieval policy that restricts the retrieval of transactions to specific nodes. Here, each transaction is given a unique retrieval tag in addition to the accompanying owner's details. This tag contains the information of the nodes that are privileged to retrieve the transaction. We defined a privileged list and stores the retrieval tag information. The purpose is to create access control and management for transaction retrieval in the blockchain network. A snippet from the smart contract shows algorithm 6 pseudocode describing the transaction verification process. The upper part of the algorithm is invoked for the submitted transaction, while the lower part is invoked for transaction retrieval.

For transaction verification to be successful, the transaction must agree with the defined format, and the owner must not mine its transaction. Also, the gas price for mining transactions must not exceed the threshold, and the number of transactions per second must not exceed the maximum. If any of these conditions fail, the transaction is dropped, and other nodes are alerted about the node's malicious behavior. The lower part of algorithm 6 describes the condition for a transaction to be retrieved. A transaction owner who needs to restrict its transaction retrieval to specific nodes publishes the privileged nodes' information to the retrieval tag that accompanies the transaction. This information is added to the privilege list. To retrieve a transaction, the requester submits its information to the blockchain network. The smart contract verifies the requester. If successful, access is allowed; else, access is denied.

Algorithm 6: Transaction Verification

Require:

Mapping(transaction => struct) public Transaction;
Mapping (transaction_format => struct) public Format
Mapping (miners => struct) Miners
Mapping (retrieval_tag => struct) public Tag

procedure: ReadTransaction (*format,price,miner,owner,requester*)
Transaction.field ← format
Transaction.gas ←gas_price
Transaction.owner ← owner
Transaction.tag ← tag
Transaction.number ← count
Transaction.requester ← []
return TRUE
end procedure

// This is invoked for a submitted transaction
procedure: TransactionVerification (*Transaction)*
require (format != NULL)
require (gas_price <= ThresholdGas)
require (owner ∄ miners)
required (Count <= ThresholdCount)
return Succeful verification with timestamp
end procedure

// This is invoked for a retrieved transaction
procedure: RetrieveTransaction (*Transaction*)
require (msg.sender ∈ Transaction.requester)
require (sender.address ∈ Tag[address])
address ← Transaction.address
Transaction.requester += msg.sender
Allow Access to Block Information
end procedure

7.2 Results

7.2.1 Detection Analysis

7.2.1.1 Outsider Threat Analysis

We evaluated the architecture's detection ability on unauthorized node transactions. A public or unauthorized node joins the blockchain network to retrieve the stored attack features or signatures. Although public nodes are part of the blockchain network, they are not privileged to participate in transaction chaining processes; hence, any attempt to get involved is considered malicious. Such transactions are discarded because they cannot be trusted. To prepare the transaction, we run the features extraction scripts on the public node. The node submits the extracted features to the blockchain network for verification and validation. Other authorized node starts mining to minimize the dissemination latency. Although other authorized nodes work to mine this transaction, we observed that a failed transaction notification is sent to the owner instead of the transaction block address. We investigated the result by checking the number of transactions in our blockchain network before and after the public node submits its transaction. It was observed that the number did not increase, indicating that the new transaction was not validated. We investigated further by manually generating the transaction address and then use it to query the Blockchain. The blockchain network did not return any transaction because there is no block with such an address in the network. The transaction failed because the owner failed the node authentication step. Based on the evaluation, we can say that the architecture allows only trusted nodes to submit transactions to the blockchain network.

7.2.1.2 Insider Threat Analysis

The insider nodes are authorized blockchain nodes that are privileged to participate in transaction validation processes. These nodes are authorized to submit, mine, and build transactions to blocks. When compromised, they can cause devastating effects on the shared information since they possess a write-access to the blockchain ledger. Here, we describe the implementation of some common insider behavioral attacks and how our architecture detects them based on the nodes' behavior. The implemented insider attacks are divided into three categories (a)those attacks targeting the Blockchain to bring it down (such as DoS), (b) the attacks that attempt to inject fake data into the database, (c) the attacks that attempt to hijack or retrieve unauthorized transaction.

7.2.1.2.1 Denial of Service Attacks

7.2.1.2.1.1 A large volume of data

We implement a case where a compromised node sends a large amount of what appears to be legitimate standard formatted transactions to mount a DoS attack on the blockchain network. This attack aims to exhaust the gas price so that a legitimate transaction will be denied validation. Authorized node prepares transactions that are a massive amount of data and submit to the blockchain network. Although other nodes are working to validate the transaction, we observed that the transactions are not mined. Notification to the owner indicates that the transaction failed due to its cost. We investigated further by manually generating the transaction address and then use it to query the Blockchain. The blockchain network did not return any transaction because no block with that transaction address resides in the network. When we check the transaction's metadata, we observed that the transaction's mining cost is greater than the gas price threshold, hence the failed notification.

7.2.1.2.1.2 Multiple Submitted Transaction

Here, a compromised blockchain node sends multiple copies of what appears to be a legitimate standard formatted request to mount a DoS attack on the blockchain network. The attack aims at exhausting the smart contract authentication and verification processes so that unverified transactions can be added to be blockchain. The node persistently submits a similar transaction to exhaust computing resources. Although other nodes attempt to start the transaction's validation process, we observed that the transactions are not attached to the blockchain because the frequency of receiving similar transactions from the same node exceeds the smart contract threshold. We continue to submit the same request from the same authorized node, and we observed that the miners stop mining after the sender was flagged to be compromised. The smart contract stops processing subsequent transactions from the same authorized node.

7.2.1.2.2 Blockchain data Injection attempt

7.2.1.2.2.1 Fake transaction values

A compromised node submits what appears to be legitimate standard formatted attack features but with fake data values. Each submitted transaction's gas price is below the threshold, and the frequency of submission is less than the threshold. Generally, it is assumed that an attacker will not hold an authorized node in a compromised state for too long due to network administrators' frequent security checks. Based on this assumption, an attacker will make all efforts to get its transactions validated to the Blockchain as quickly as possible. The transaction owner submits and mines its transaction to get it validated as fast as possible. The result showed that the transaction is not attached to the blockchain network, although other authorized nodes are also mining to validate the same transaction. The smart contract drops the transaction because the owner attempts to mine its transaction, making the activity flagged as malicious. The transaction failed because the owner attempts to mine its transaction, which is classified as malicious behavior.

7.2.1.2.3 Unauthorized Transaction Retrieval Attempt

7.2.1.2.3.1 Transaction Access Control

Here, a public node attempts to retrieve an unauthorized transaction. We prepared a transaction that the retrieval is restricted to the authorized nodes only. This transaction was submitted and mined to the blockchain network. A public node retrieved the block address and queried the blockchain for the content of a newly added block. The result showed that no information was returned because the node is not privileged to retrieve the data. The smart contract invokes the code that compares the information of the requesting public node to what is listed in the privileged list. Based on this, the smart contract controls access to that transaction. Access to the newly added block content is denied because the requesting node is not part of the authorized node. We investigate further by querying the Blockchain using another authorized node. We observed that the node successfully downloads the block's content from the Blockchain.

7.2.1.2.3.2 Hijacking validating transaction

Here, a malicious node attempts to copy the information of a block during the validation period. We develop and run a script on the malicious node that can replicate information. This attack aims to hijack the block's content the node is working on before validation is complete. We observed that instead of returning the validating block's content, the script returns the block's hash. The block's content was invisible to the miner because it has been sealed with a lock (hash) by the blockchain protocol. Based on these results, we can conclude that the architecture can provide a server-side authentication and verification process to detect and prevent malicious activities on the stored attack information.

7.2.2. Performance Analysis

7.2.2.1 Detection Time

We evaluated the detection time for all attacks perform on the blockchain network. The detection time is defined as the time it takes for the architecture to notify a transaction's failure. This time is measured from when the transaction is submitted to the time the failure notification is received. Both the submission and notification times are timestamped and recorded for each attack. Fig.17 shows the difference in the detection time for all attacks implemented. We observed that the time taken to detect a DoS attack with multiple transactions is the highest. The result could be because the algorithm waits to count the number of transactions per sec from the same sender before sending a notification.

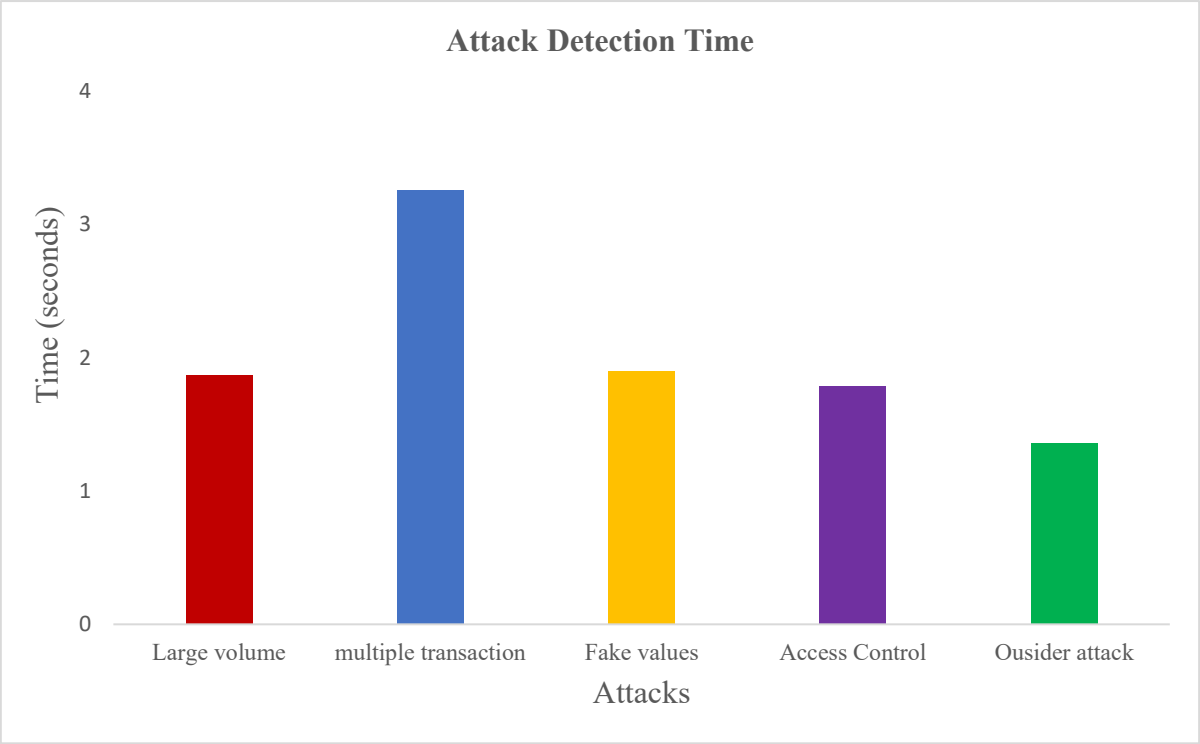

Fig. 17 The attack detection time

We described the proposed architecture attributes and compared them with other approaches highlighted in the related work (Table X). The table shows the distinction between our proposed approach and other methods used for cooperative intrusion detection systems.

Table X: The proposed architecture's attributes

Properties	**Trust-chain [49]**	**CBSigIDS [47]**	**CIOTA [41]**	**SectNet [48]**	**BAD [40]**	**DOMINO [44]**	**Our Method**
Data Sharing	✓	✓	x	✓	x	✓	✓
Blockchain	✓	✓	✓	✓	✓	x	✓
Detect External threats	x	✓	✓	✓	x	✓	✓
Detect Insider Threats	✓	x	x	x	x	x	✓
Compatible with different IDS	x	✓	✓	x	x	✓	✓
Transaction verification	x	x	x	x	x	x	✓
Useful for both Signature and Anomaly attack features	✓	x	x	x	x	✓	✓

CHAPTER EIGHT

APPLICATION IN HEALTHCARE SYSTEMS

8.1 Introduction

Healthcare industries are proposing interoperability among different healthcare providers to strengthen healthcare delivery. The interoperability is important because a patient's diagnosis and treatment journey can take them from a physician's office to an imaging center or the hospital's operating room. Each stop generates a record, such as doctor's notes, test results, medical device data, discharge summaries, or information essential to the social determinants of health, which become part of a patient's electronic health record in each setting[68]. For the best outcome, this health information should be accessible and securely exchanged among all sources that accompany the patient's treatment every step of the way. This interoperability will strengthen care coordination and improve safety, quality, efficiency, and encouragement of robust health registries[69]. Since interoperability requires that shared patient information is of high integrity, confidential and consistent, healthcare systems need a tamperproof method of exchanging health records. Research methods proposed encryption and a centralized database to share patient health records securely [21, 57]. Although these solutions look promising and straightforward, some of the major problems threatening them are summarized below.

8.2 Problem Statements

The significant problems facing the available solution are

- Encryptions only ensure the confidentiality of the data. It does not guarantee integrity and consistency.
- The medium of exchange can be hacked, thereby compromising the integrity and consistency of the shared data.
- The database housing the Electronic Health Records (EHRs) makes the data susceptible to single-point-of-failure or man-in-the-middle attacks.

- The lack of a universal format for EHRs exchange makes it difficult to detect and prevent malicious activities.

Several research studies have proposed blockchain applications for securing and sharing personal health data [55-60]. Despite the vast number of blockchain applications in healthcare, little attention has been focused on inter-healthcare health record exchange, this application's motivation. This chapter describes the application of the proposed architecture to secure EHRs exchange across different healthcare systems. The healthcare application, an extension to the proposed architecture, uses the architecture concept to exchange patients' health history between different healthcare systems. A healthcare system builds and maintains patient health information on an individual private blockchain platform that works on the same principle described for our proposed architecture. This application focuses on the interaction of blockchain networks to exchange patients' EHRs across different healthcare providers. The application describes a novel method of data sharing between different blockchain networks using deployed smart contracts. Figure 18 shows the pictorial representation of the proposed solution.

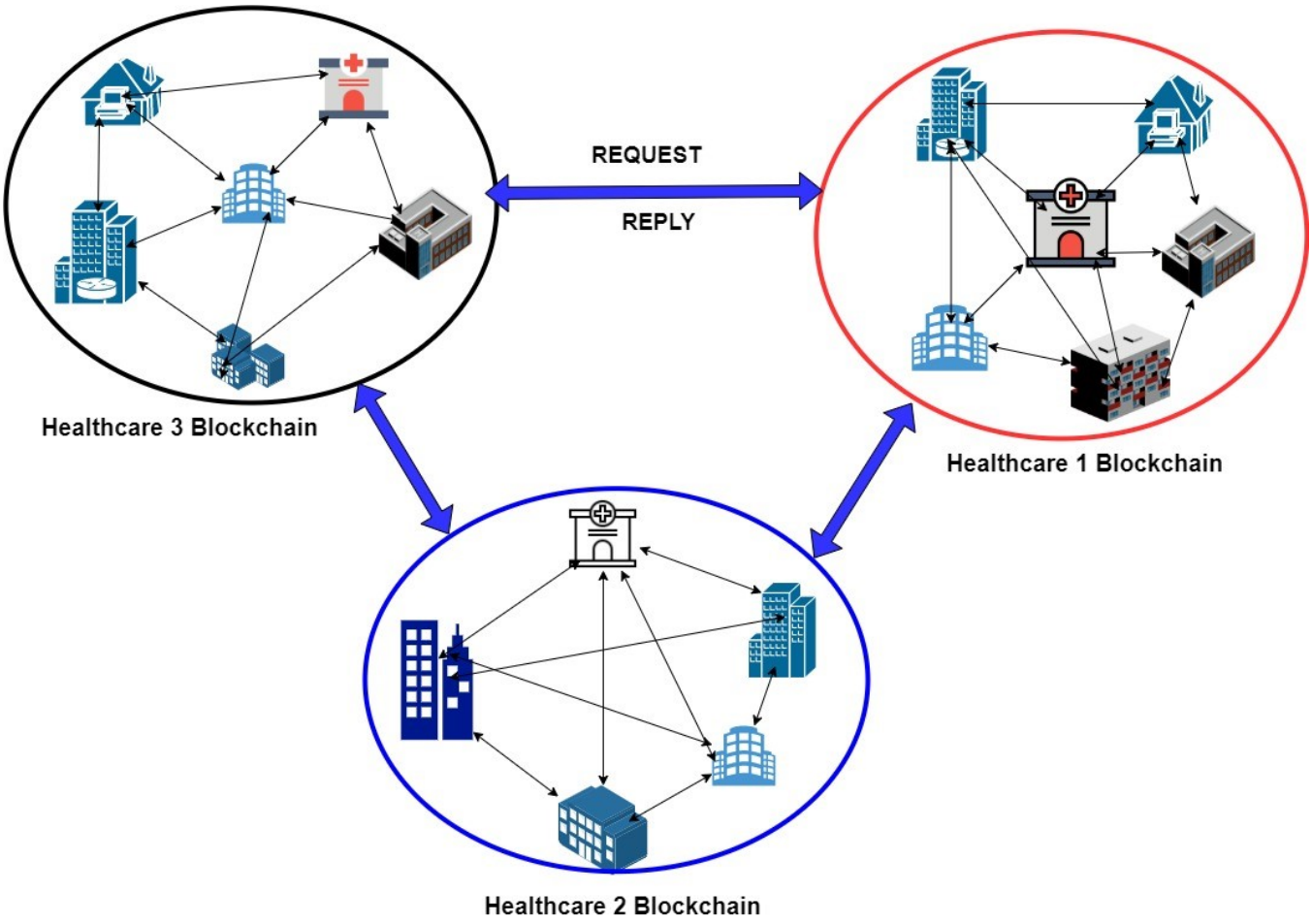

Fig. 18 The pictorial representation of the e-health application

8.3 Methodology

The work describes the inter-healthcare request and reply formation, verification, and validation. The building block is divided into three prominent stages, as shown in Figure 19.

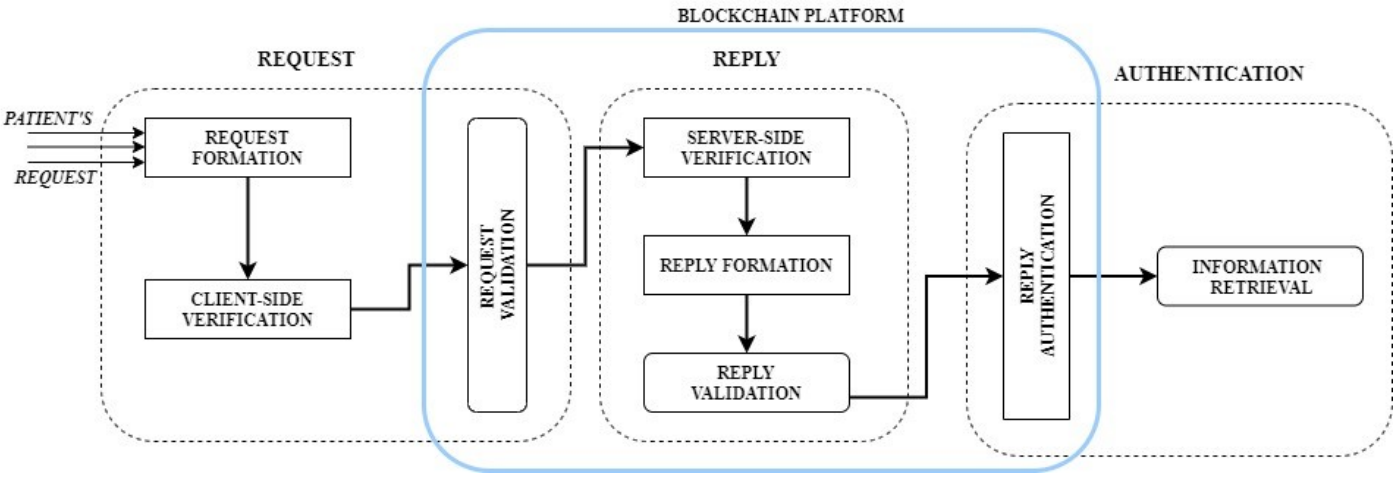

Fig. 19 The Building blocks of the e-health application

8.3.1 Request

The request stage is subdivided into three categories: request formation, client-side verification, and request validation. In this context, we define a transaction as any of the following:

- A patient's health information about to be stored in the blockchain network
- A request sent to another healthcare systems for patients' medical history
- A reply that carries the requested patient's information

The request is formed to obtain patients' medical history from another healthcare system or medical test results from another laboratory. For example, a student who lives, say in New York, the USA travels to say London, UK. If the student had to visit a hospital for treatment in the visiting country, the medical history should be retrieved from the home hospital for best medical care. In this case, we assume that all healthcare systems run private blockchain networks to store patient's medical information securely. A patient's medical history can be retrieved by preparing a request with a piece of information unique to the patient. During the process, a requester (doctor or nurse in the visiting hospital) fills the required information into a developed script running on the nodes/workstations. This script captures the patients' necessary information such as name, date of birth, social security (SSN), name of the home hospital, and requester's unique code. Here, the doctor/nurse's workstation is part of that hospital's blockchain nodes.

Through the script, the node performs a client-side verification of the information and the requester's network.

The request is developed into an agreed-upon formatted transaction, digitally signed by the node, and submitted to the hospital's private blockchain network. Apart from the submitted transaction, the node submits a transaction tag that contains the sending node's information and the requester's unique code. The smart contract that runs on the blockchain verifies the transaction's format and authenticates the node using the transaction tag. The purpose is to ensure that the transaction originates from privileged nodes and authorized personnel. Algorithm 7 pseudocode shows the snippet of the smart contract that handles the verification process. Since the healthcare blockchains communicate via smart contracts, each smart contract running on the blockchain networks contains the processes described in algorithm 7.

The smart contract verifies the transaction and counts the number of requests from the same node within a certain period. We set a policy that restricts the processing of more than a transaction from the same node in a period. The code also confirms the destination healthcare in the lookup table and verifies the origin of the submitted transaction using the node's digital signature. It checks the requester and node's privilege of submitting transactions by comparing their information to the authorized list. Furthermore, the smart contract ensures transaction retrieval access control to patients' information requester. For a request to be successful, it must agree with the format, and the number of requests within a certain period must not exceed the threshold. Also, the requester must be authorized, the destination healthcare blockchain information must be available in the lookup table, and requesting node must be privileged. If any of these verification steps fail, the transaction is dropped. A successful request is validated and attached to the blockchain, as explained in the previous chapters.

Algorithm 7: Request Verification

```
procedure: ReadRequest (Transaction, Tag )
        Transaction.field ← field
        Transaction.number ← count
        Tag.pubkeys ← pubkey
        Transaction.owner ← owner
        Tag.codes ← identifier
        Tag.networkAddrs ← destdddr
        Tag.accounts ← account
        Transaction.data ← requestinfo
        return TRUE
end procedure

// This is invoked for a submitted request
procedure: RequestVerification (Request)
        require (field != NULL)
        require (count <= Threshold)
        require (pubkey ∈ Public_keys)
        require ( code ∈ Identifiers)
        require (destaddr ∈ Networks)
        require (account ∈ Account)
        require(pubkey verifies priv_key)
        return Successful verification with timestamp
end procedure

// This is invoked to authenticate the imcoming reply
procedure: ReadReply (Reply )
        Reply.addr ← srcaddr
        Reply.data← info
        Reply.network ← networkinfo
        return TRUE
end procedure

procedure: ReplyAuthentication (Reply)
        require(srcaddr == destaddr)
        require (info == requestinfo)
        require(networkinfo ∈ Networks)
        Validate and allow access to Requesting node
end procedure
```

8.3.2 Reply

After a successful validation process, the smart contract routes the request to the designated healthcare blockchain based on the name to address mapping on the lookup table. The destination smart contract

verifies the format of the incoming request and the requesting blockchain. Algorithm 8 shows a pseudocode that describes the verification of an incoming request. The source information, request format, and digital signature of the source blockchain are verified. The supplied and requested information about the patient are confirmed. For an incoming request to be successful, the request's format must agree with the standard, source information must pass the verification step, requested patient’s information must be available in the destination healthcare network, and the source public key must verify its private key. If any verification steps fail, the request is dropped, and a failed request is sent to the source network. A successful request is validated and attached to the blockchain. Based on the required information, a reply is prepared and submitted for verification. The transaction is verified, as explained in previous chapters. A successfully verified reply is validated and routed back to the source blockchain network.

Algorithm 8: Reply Formation

```
procedure: ReadIncomingRequest (Request )
        Request.field ← field
        Request.number ← count
        Request.description ← information
        Request.owner ← Patient_information
        Tag.address ← address
        Tag.account ← account
        return TRUE
end procedure

// This is invoked for an incoming request
procedure: IncomingRequestVerification (Incoming Request)
        require (field != NULL)
        require (count <= Threshold)
        require (information ∈ blockchain metadata)
        require (address ∈ Networks)
        require (account ∈ Account)
        require(pubkey verifies priv_key)
        return Successful verification with timestamp
end procedure

// This is invoked for a reply formation, verification and validation
procedure: ReplyFormation (Reply)
        require (field != NULL)
        require (pubkey ∈ Public_keys)
        require (address ∈ Networks)
        require (account ∈ Accounts)
        require (data == patient_information)
        Return validated reply
end procedure
```

8.3.3 Authentication

The source network verifies the incoming reply. The smart contract verifies the incoming reply, the address of the reply's network, and its digital signature, as shown in the lower part of algorithm 7. If the verification process is successful, the transaction (reply) is validated and attached to the blockchain. The blockchain is updated, and the newly added block reflects on the ledger of every node in the network. All blockchain nodes receive the newly added block notification but do not have access to the block's content.

The requesting node retrieves the information in the block, and a developed script converts it to a format that the requester (healthcare worker) can easily understand.

8.4 RESULTS

In this ongoing work, we set up the proposed application in the laboratory. We set up two different private blockchain networks (I and II), with each network comprises three nodes. For each blockchain network, we use Solidity *v 0.6.2* implementation for smart contracts and *geth v 1.9.0* for Ethereum. The smart contract is written as described above and mined into the blockchain network. A transaction (request) was prepared as explained and submitted to the transaction in blockchain network I. We randomly generate ten-string long numbers to serve as the unique requester code. The miner information, the request format and reply, and the requester's unique codes are written in the smart contract. Apart from this, information about blockchain II is written in the smart contract. This smart contract is mined to the blockchain I network. We developed a similar smart contract (with the blockchain I information) and mined it into blockchain II's network.

We evaluate a preliminary result on the security analysis of the proposed system. We anticipated and tested ways a compromised node could access unauthorized patient information within a healthcare blockchain network. We performed the attack and presented the result obtained from our architecture. We showed how the architecture detects a compromised node. We compare the proposed system's attributes with other similar works from the literature, and we present the result in table XI.

8.4.1 Security Analysis

8.4.1.1 *Multiple Submitted Requests*

Here, a compromised blockchain node sends multiple copies of what appears to be a legitimate standard formatted request to mount a DoS attack on the blockchain network. The node persistently submits a similar request transaction to exhaust computing resources. Although other nodes attempt to start the transaction's validation process, we observed that the transactions are not attached to the blockchain

because the frequency of receiving similar transactions from the same node exceeds the threshold. We continue to submit the same request from the same authorized node, and we observed that the miners stop mining after the sender was flagged to be compromised. The smart contract stops processing subsequent transactions from the same authorized node.

8.4.1.2 *Unauthorized retrieval of patient's record*

Here, a node attempts to retrieve an unauthorized transaction. The node queries the Blockchain for the content of a newly added block in which it is not the requester. The result showed that no information was returned because the node is not privy to retrieve the data. In the smart contract, we define a policy that limits information retrieval to the node that submitted the request. Based on this policy, the smart contract controls access to that transaction block. We investigate further by querying the Blockchain using the requester. We observed that the node successfully downloads the block's content from the Blockchain.

Table XI: The proposed architecture's attributes

Properties	**Mahore [55]**	**i-Blockchain [65]**	**Yang [70]**	**MedRec [58]**	**Carter [59]**	**A.Zhang [69]**	**Amofa [62]**	**Our System**
Blockchain-based	✓	✓	x	✓	✓	✓	✓	✓
Access Control	✓	✓	✓	✓	✓	✓	✓	✓
Privacy Preservation	✓	✓	✓	✓	✓	✓	✓	✓
Inter-healthcare PHI exchange	x	x	x	x	✓	x	x	✓
Information Verification	x	x	x	x	x	x	x	✓

CHAPTER NINE

CONCLUSION

This dissertation proposed an architecture that enhances the detection of coordinated or distributed cyberattacks to strengthen cooperative intrusion detection systems. The solution, a permissionless public-private blockchain network, leverages blockchain network capabilities to secure cyberattack features and signatures and promptly share them among participating IDS nodes in real-time. It is a public blockchain because it allows a permissionless joining and leaving of nodes, whereas it is considered a private blockchain because only specific nodes are allowed to participate in the transaction validation processes. The solution's novelties include; a) detecting and preventing malicious activities on cyberattack features and signatures from public nodes, b) facilitating scalable and secured attack features and signature exchange among IDS nodes in computer networks, c) continuously monitoring the activities of blockchain nodes to detect compromised insider nodes, d) providing strong server-side authentication and verification process for transaction owners and submitted transaction, e) presenting a standard format for verified cyberattack features and signature which encourages heterogeneous participation of IDS nodes, f) storing the verified transaction in a tamper-proof, immutable, distributive ledger and g) robustness to public nodes joining and leaving the network in real-time.

We first implement the architecture in the laboratory, then deployed it to a google cloud platform. We compared the results from the two implementations using some performance metrics such as transaction dissemination latency and scalability. We further carry out the security analysis. The security analysis investigates how the architecture detects some typical insider and outsider attacks directed towards contaminating the stored information. The result shows that the architecture can detect an attack within a short period of launching the attack and prevent malicious activities from insider and outsider threats on shared data. At the same time, the performance result shows that the architecture's latency is low.

Furthermore, we describe the architecture's application in the healthcare system. In the ongoing work, we describe how the application is suitable for exchanging patient's health records across different healthcare

systems. We evaluated the performance using security analysis and detection time metrics. The preliminary result shows that the proposed solution can detect malicious activities on the patient records.

Publications and Patent Disclosures

1. **O. Ajayi** and T. Saadawi, **“** Blockchain-based Architecture for Cooperative Intrusion Detection System” Submitted to Elsevier Journal of Parallel and Distributed Computing. Jan 2021.
2. **O. Ajayi** and T. Saadawi, “Detecting Insider attacks in Blockchain Networks” accepted for oral presentation at the 2021 International Symposium on Networks, Computers and Communications: Trust, Security, and Privacy (ISNCC-TSP), October 31 - November 2, 2021. Dubai UAE.
3. **O. Ajayi**, M. Abouali and T. Saadawi, “ Blockchain Architecture for Secured Inter-Healthcare Electronic Health Records Exchange” 12th International workshop on information Network and Design (WIND 2020), August 31- September 2, 2020. University of Victoria, Victoria, Canada.
4. **O. Ajayi,** M. Abouali and T. Saadawi, "Secured Inter-Healthcare Patient Health Records Exchange Architecture*,″* 2020 IEEE International Conference on Blockchain (Blockchain)*,* Rhodes, Greece, 2020, pp. 456-461, doi: 10.1109/Blockchain50366.2020.0006
5. **O. Ajayi** and T. Saadawi, "Blockchain-Based Architecture for Secured Cyber-Attack Features Exchange" 7th International Conference on Cyber Security and cloud computing (IEEE CSCLOUD 2020), August 1st-3rd, 2020, New York, USA
6. **O. Ajayi** and T. Sadaawi, "BACOIDS: A blockchain Architecture for Data Security Enhancement in Cooperative Intrusion Detection System" 46th Annual Conference of Association of Egyptian-American Scholar (AEAS 2019), Dec. 24-26, 2019, Cairo, Egypt.
7. **O. Ajayi** and T. Sadaawi**,** "Blockchain-Based Architecture for Secured Data Storage" Chemical and Biological Defense Science & Technology Conference (CBD S&T 2019), Nov 17-21, 2019, Cincinnati, Ohio, USA.
8. **O. Ajayi,** O. Igbe**,** and T. Sadaawi**,** "Consortium Blockchain-Based Architecture for Cyber-attack Signatures and Features Distribution" 2019 IEEE 10th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON 2019), Oct 10th – 12th, 2019, Columbia University, New York, USA.

9. **O. Ajayi**, M. Cherian, and T. Sadaawi**,** ″Secured Cyber-Attack Signatures Distribution using Blockchain Technology″ 17th IEEE International Conference on Embedded and Ubiquitous Computing (IEEE EUC 2019) August 1–3, 2019, New York, USA **.**

10. O. Igbe, **O. Ajayi**, and T. Saadawi, "Detecting Denial of Service attacks using a combination of Dendritic Cell Algorithm(DCA) and Negative Selection Algorithm(NSA)" 2nd International Conference on Smart Cloud (Smart Cloud 2017) Nov 3rd-5th, 2017, New York, USA.
11. O. Igbe, **O. Ajayi**, and T. Saadawi, **"**Denial of Service attacks detection using Dendritic Cell Algorithm(DCA)" 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON 2017) Oct 19th – 21st 2017, Columbia University, New York, USA

Patent Disclosures

1. **O. Ajayi,** O. Igbe and T. Sadaawi**,** A patent on "**Secured Blockchain Architecture for Cyberattack Signatures and Features Distribution**" Summitted to the new Technology Disclosure, Technology Commercialization Office, The City University of New York, 555 West 57th street, suite 1407, New York, NY 10019 on June 10, 2019, final submission 19A0006 RE: Provisional Application 62/741,276 | Our Ref No.: 03284.0226US01
2. **O. Ajayi,** M. Abouali and T. Sadaawi**,** A patent disclosure on " **Blockchain Architecture for Secured Inter-Healthcare Electronic Health Records Exchange** " Summitted to the new Technology Disclosure, Technology Commercialization Office, The City University of New York, 555 West 57th street, suite 1407, New York, NY 10019 on July 07, 2020

References

[1] H. Yin, D. Guo, K. Wang, Z. Jiang, Y. Lyu, and J. Xing, ‘‘Hyperconnected network: A decentralized trusted computing and networking paradigm,’’ IEEE Netw., vol. 32, no. 1, pp. 112–117, Jan./Feb. 2018.

[2] K. Fan, W. Jiang, H. Li, and Y. Yang, ‘‘Lightweight RFID protocol for medical privacy protection in IoT,’’ IEEE Trans Ind. Informat., vol. 14, no. 4, pp. 1656–1665, Apr. 2018.

[3] T. Chajed, J. Gjengset, J. Van Den Hooff, M. F. Kaashoek, J. Mickens, R. Morris, and N. Zeldovich, ‘‘Amber: Decoupling user data from Web applications,’’ in Proc. 15th Workshop Hot Topics Oper. Syst. (HotOS XV), Warth-Weiningen, Switzerland, 2015, pp. 1–6.

[4] C. Perera, R. Ranjan, and L. Wang, ‘‘End-to-end privacy for open big data markets,’’ IEEE Cloud Comput., vol. 2, no. 4, pp. 44–53, Apr. 2015.

[5] X. Zheng, Z. Cai, and Y. Li, ‘‘Data linkage in smart Internet of Things systems: A consideration from a privacy perspective,’’ IEEE Commun. Mag., vol. 56, no. 9, pp. 55–61, Sep. 2018.

[6] Y. Liang, Z. Cai, J. Yu, Q. Han, and Y. Li, ‘‘Deep learning based inference of private information using embedded sensors in smart devices’’ IEEE Netw. Mag., vol. 32, no. 4, pp. 8–14, Jul./Aug. 2018.

[7] https://docs.openstack.org/, accessed on 2021-02-08.

[8] S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas, “Modeling intrusion detection system using hybrid intelligent systems,” Journal of network and computer applications, vol. 30, no. 1, pp. 114–132, 2007..

[9] O. Igbe, O. Ajayi, and T. Saadawi, “Denial of Service Attack Detection using Dendritic Cell Algorithm” 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON 2017) Oct 19th – 21st 2017, Columbia University, New York, USA.

[10] O. Igbe, O. Ajayi, and T. Saadawi, “Detecting Denial of Service attacks using a combination of Dendritic Cell Algorithm(DCA) and Negative Selection Algorithm(NSA)” 2nd International conference on Smart Cloud (Smart Cloud 2017) Nov 3rd-5th, 2017, New York, USA.

[11] F. Gong, ‘‘Next generation intrusion detection systems (IDS),’’ McAfee Netw. Secur. Technol. Group, Santa Clara, CA, USA, White Paper, 2003

[12] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang and J. Han, "When Intrusion Detection Meets Blockchain Technology: A Review," in IEEE Access, vol. 6, pp. 10179-10188, 2018, doi: 10.1109/ACCESS.2018.2799854.]

[13] M. Kumar and A. K. Singh, "Distributed Intrusion Detection System using Blockchain and Cloud Computing Infrastructure," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 248-252, doi: 10.1109/ICOEI48184.2020.9142954.

[14] Y. L. Dong, J. Qian, M. L. Shi, “A cooperative intrusion detection system based on autonomous agents,” IEEE CCECE 2003, Vol. 2, pp. 861– 863, 2003.

[15] C. C. Lo, C. Huang, J. Ku, A cooperative intrusion detection system framework for cloud computing networks, in: In: Proceedings of the 2010 39th International Conference on Parallel Processing Workshops,ICPPW '10, 2010, pp. 280-284.

[16] Y.-S. Wu, B. Foo, Y. Mei, and S. Bagchi, ‘‘Collaborative intrusion detection system (CIDS): A framework for accurate and efficient IDS,’’ in Proc. Annu. Comput. Secur. Appl. Conf. (ACSAC), Dec. 2003, pp. 234–244.

[7] O. Ajayi, M. Cherian and T. Saadawi, "Secured Cyber-Attack Signatures Distribution using Blockchain Technology," 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), New York, NY, USA, 2019, pp. 482-488, doi: 10.1109/CSE/EUC.2019.00095.

[18] W. Zhang, S. Teng, H. Zhu, D. Liu, "A Cooperative Intrusion Detection Model Based on Granular Computing and Agent Technologies", *J. International Journal of Agent Technologies and Systems,* vol. 5, no. 3, pp. 54-74, 2013

[19] S.R. Snapp, J. Brentano, GV dias, T.L. Goan, L.T. Heberlein, C. Ho, K.N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, D.M. Teal, and D. Mansur. DIDS (distributed intrusion detection system) — motivation, architecture, and an early prototype. In Proceedings of the 14th National Computer Security Conference, pages 167–176, October 1991.

[20] M. Uddin, A. Abdul Rehman, N. Uddin, J. Memon, R. Alsaqour, and S. Kazi, "Signature-based Multi-Layer Distributed Intrusion Detection" International Journal of Network Security, Vol.15, No.2, PP.97-105, Mar. 2013

[21] Sultan Aldossary, William Allen. Data Security, Privacy, Availability and Integrity in Cloud Computing: Issues and Current Solutions. (IJACSA) International Journal of Advanced Computer Science and Applications,Vol. 7, No. 4, 2016 pp.485-498

[22] C. Wang, S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," Computers, IEEE Transactions on, vol. 62, no. 2, pp. 362–375, Feb 2013

[23] G.D. Kurundkar, N.A. Naik, and S.D. Khamitkar, "Network Intrusion Detection using SNORT" International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 2,Mar-Apr 2012, pp.1288-1296

[24] K. Sultan, U. Ruhi, and R. Lakhani, "Conceptualizing Blockchains: Characteristics and Applications," in 11th IADIS International Conference on Information Systems, 2018, pp. 49–57.

[25] M. Tanrıverdi and A. Tekerek, "Implementation of Blockchain Based Distributed Web Attack Detection Application," 2019 1st International Informatics and Software Engineering Conference (UBMYK), Ankara, Turkey, 2019, pp. 1-6, doi: 10.1109/UBMYK48245.2019.8965446

[26] S. Nakamoto (2008) Bitcoin: a peer-to-peer electronic cash http://bitcoin.org/bitcoin.pdf

[27] I. Weber et al., "On Availability for Blockchain-Based Systems," 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), Hong Kong, 2017, pp. 64-73, doi: 10.1109/SRDS.2017.15.

[28] T. Ahram, A. Sargolzaei, S. Sargolzaei, J. Daniels and B. Amaba, "Blockchain technology innovations," 2017 IEEE Technology & Engineering Management Conference (TEMSCON), Santa Clara, CA, 2017, pp. 137-141, doi: 10.1109/TEMSCON.2017.7998367.

[29] D. Patel, J. Bothra and V. Patel, "Blockchain exhumed," 2017 ISEA Asia Security and Privacy (ISEASP), Surat, 2017, pp. 1-12, doi: 10.1109/ISEASP.2017.7976993.

[30] T. M. Fernández-Caramés and P. Fraga-Lamas, "A Review on the Application of Blockchain to the Next Generation of Cybersecure Industry 4.0 Smart Factories," in IEEE Access, vol. 7, pp. 45201-45218, 2019, doi: 10.1109/ACCESS.2019.2908780.

[31] X. Zheng, R. R. Mukkamala, R. Vatrapu and J. Ordieres-Mere, "Blockchain-based Personal Health Data Sharing System Using Cloud Storage," 2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom), Ostrava, 2018, pp. 1-6, doi: 10.1109/HealthCom.2018.8531125.

[32] X. Yang, T. Li, R. Liu and M. Wang, "Blockchain-Based Secure and Searchable EHR Sharing Scheme," 2019 4th International Conference on Mechanical, Control and Computer Engineering (ICMCCE), Hohhot, China, 2019, pp. 822-8223, doi: 10.1109/ICMCCE48743.2019.00188.

[33] S. Amofa et al., "A Blockchain-based Architecture Framework for Secure Sharing of Personal Health Data," 2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom), Ostrava, 2018, pp. 1-6, doi: 10.1109/HealthCom.2018.8531160.

[34] K. Ito, K. Tago and Q. Jin, "i-Blockchain: A Blockchain-Empowered Individual-Centric Framework for Privacy-Preserved Use of Personal Health Data," 2018 9th International Conference on Information Technology in Medicine and Education (ITME), Hangzhou, 2018, pp. 829-833, doi: 10.1109/ITME.2018.00186.

[35] G. Yang and C. Li, "A Design of Blockchain-Based Architecture for the Security of Electronic Health Record (EHR) Systems," 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Nicosia, 2018, pp. 261-265, doi: 10.1109/CloudCom2018.2018.00058.

[36] P. Zhang, M. A. Walker, J. White, D. C. Schmidt and G. Lenz, "Metrics for assessing blockchain-based healthcare decentralized apps," *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom),* Dalian, 2017, pp. 1-4, doi: 10.1109/HealthCom.2017.8210842

[37] X. Liang, J. Zhao, S. Shetty, J. Liu and D. Li, "Integrating blockchain for data sharing and collaboration in mobile healthcare applications," 2017 IEEE 28th Annual International Symposium on Personal, Indoor

[38] Sultan Aldossary, William Allen. Data Security, Privacy, "Availability and Integrity in Cloud Computing: Issues and Current Solutions.", (IJACSA) International Journal of Advanced Computer Science and Applications,Vol. 7, No. 4, 2016 pp.485-498.

[39] Schär, F. "*Blockchain Forks: A Formal Classification Framework and Persistency Analysis*;" World Scientific Europe: London, UK, 2020

[40] M Signorini and M Pontecorvi, W Kanoun, and R Di Pietro, "BAD: a Blockchain Anomaly Detection solution" arXiv:1807.03833v2, [cs. C.R.] 12 jul 2018

[41] T. Golomb, Y. Mirsky and Y. Elovici " CIoTA: Collaborative IoT Anomaly Detection via Blockchain" arXiv:1803.03807v2, [cs.CY] 09 Apr 2018

[42] Gu, J, B Sun, X Du, J Wang, Y Zhuang and Z Wang (2018). Consortium blockchain-based malware detection in mobile devices. IEEE Access, 6, 12118–12128.

[43] S.R. Snapp, J. Brentano, G.V. dias, T.L. Goan, L.T. Heberlein, C. Ho, K.N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, D.M. Teal, and D. Mansur. DIDS (distributed intrusion detection system)

— motivation, architecture, and an early prototype. In Proceedings of the 14th National Computer Security Conference, pages 167–176, October 1991.

[44] V. Yegneswaran, P. Barford, S. Jha, "Global intrusion detection in the DOMINO overlay system", *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS),* pp. 1-17, 2004.

[45] Y. L. Dong, J. Qian, M. L. Shi, “A cooperative intrusion detection system based on autonomous agents,” IEEE CCECE 2003, Vol. 2, pp. 861– 863, 2003.

[46] C. C. Lo, C. Huang, J. Ku, “A cooperative intrusion detection system framework for cloud computing networks,” In Proceedings of the 2010 39th International Conference on Parallel Processing Workshops,ICPPW '10, 2010, pp. 280-284.

[47] S. Tug, W. Meng and Y. Wang, "CBSigIDS: Towards Collaborative Blockchained Signature-Based Intrusion Detection," 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 2018, pp. 1228-1235, doi: 10.1109/Cybermatics_2018.2018.00217.

[48] K. Wang, J. Dong, Y. Wang and H. Yin, "Securing Data With Blockchain and AI," in IEEE Access, vol. 7, pp. 77981-77989, 2019, doi: 10.1109/ACCESS.2019.2921555.

[49] N. Kolokotronis, S. Brotsis, G. Germanos, C. Vassilakis and S. Shiaeles, "On Blockchain Architectures for Trust-Based Collaborative Intrusion Detection," 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 2019, pp. 21-28, doi: 10.1109/SERVICES.2019.00019.

[50] S. Homayoun, A. Dehghantanha, R. M. Parizi and K. R. Choo, "A Blockchain-based Framework for Detecting Malicious Mobile Applications in App Stores," 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), Edmonton, AB, Canada, 2019, pp. 1-4, doi: 10.1109/CCECE.2019.8861782.

[51] N. A. Dawit, S. S. Mathew and K. Hayawi, "Suitability of Blockchain for Collaborative Intrusion Detection Systems," 2020 12th Annual Undergraduate Research Conference on Applied

Computing (URC), Dubai, United Arab Emirates, 2020, pp. 1-6, doi: 10.1109/URC49805.2020.9099189.

[52] M. Kumar and A. K. Singh, "Distributed Intrusion Detection System using Blockchain and Cloud Computing Infrastructure," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 248-252, doi: 10.1109/ICOEI48184.2020.9142954.

[53] N. A. Dawit, S. S. Mathew and K. Hayawi, "Suitability of Blockchain for Collaborative Intrusion Detection Systems," 2020 12th Annual Undergraduate Research Conference on Applied Computing (URC), Dubai, United Arab Emirates, 2020, pp. 1-6, doi: 10.1109/URC49805.2020.9099189.

[54] B. Jia and Y. Liang, "Anti-D chain: A lightweight DDoS attack detection scheme based on heterogeneous ensemble learning in blockchain," in China Communications, vol. 17, no. 9, pp. 11-24, Sept. 2020, doi: 10.23919/JCC.2020.09.002.

[55] V. Mahore, P. Aggarwal, N. Andola, Raghav and S. Venkatesan, "Secure and Privacy Focused Electronic Health Record Management System using Permissioned Blockchain," 2019 IEEE Conference on Information and Communication Technology, Allahabad, India, 2019, pp. 1-6, doi: 10.1109/CICT48419.2019.9066204.

[56] K. Ito, K. Tago and Q. Jin, "i-Blockchain: A Blockchain-Empowered Individual-Centric Framework for Privacy-Preserved Use of Personal Health Data," *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, Hangzhou, 2018, pp. 829-833, doi: 10.1109/ITME.2018.00186.

[57] X. Zheng, R. R. Mukkamala, R. Vatrapu and J. Ordieres-Mere, "Blockchain-based Personal Health Data Sharing System Using Cloud Storage," *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*, Ostrava, 2018, pp. 1-6, doi: 10.1109/HealthCom.2018.8531125.

[58] A. Azaria, A. Ekblaw, T. Vieira and A. Lippman, "MedRec: Using Blockchain for Medical Data Access and Permission Management," *2016 2nd International Conference on Open and Big Data (OBD)*, Vienna, 2016, pp. 25-30, doi: 10.1109/OBD.2016.11.

[59] G. Carter, H. Shahriar and S. Sneha, "Blockchain-Based Interoperable Electronic Health Record Sharing Framework," 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 2019, pp. 452-457, doi: 10.1109/COMPSAC.2019.10248.

[60] Z. Shae and J. J. P. Tsai, "On the Design of a Blockchain Platform for Clinical Trial and Precision Medicine," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, 2017, pp. 1972-1980, doi: 10.1109/ICDCS.2017.61.

[61] P. Zhang, M. A. Walker, J. White, D. C. Schmidt and G. Lenz, "Metrics for assessing blockchain-based healthcare decentralized apps," 2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom), Dalian, 2017, pp. 1-4, doi: 10.1109/HealthCom.2017.8210842.

[62] S. Amofa et al., "A Blockchain-based Architecture Framework for Secure Sharing of Personal Health Data," 2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom), Ostrava, 2018, pp. 1-6, doi: 10.1109/HealthCom.2018.8531160.

[63] Zhang, P., Walker, M., White, J., Schmidt, D.C., and Lenz, G.: 'Metrics for assessing Blockchain-based healthcare decentralized apps', Proceedings of 2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom), October 12-15, 2017, Dalian, China.

[64] Ingo Weber, Vincent Gramoli, Mark Staples, Alex Ponomarev, Ralph Holz, An Binh Tran, and Paul Rimba. 2017. On Availability for Blockchain-Based Systems. In SRDS'17: IEEE International Symposium on Reliable Distributed Systems

[65] V. Paxson. “Bro: a system for detecting network intruders in real time. “Computer Networks, 31(23-24), December 1999.

[66] R. McRee, “ Suricata: An Introduction” Information Systems Security Association Journal, USA. August 2010

[67] L. Dhanabal, S.P. Shantharajah, A study on NSL-KDD dataset for intrusion detection system based on classification algorithms, International Journal of Advanced Research in Computer and Communication Engineering 4 (2015) 446–452

[68] Liu V, Musen MA, Chou T. Data breaches of protected health information in the United States. JAMA. Vol. 313, num 14, (2015) pp 1471–1473.

[69] Wang CJ, Huang DJ. The HIPAA conundrum in the era of mobile health and communications. JAMA. Vol. 310, num 11, (2013) Pp.1121–1122.

[70] Yang, Y., and Ma, M., ”Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-Health clouds”, IEEE Transactions on Information Forensics and Security, vol. 11, no. 4, pp. 746-759, 2016.

[71] L. A. Ajao, J. Agajo, E. A. Adedokun, and L. Karngong, ‘‘Crypto hash algorithm-based blockchain technology for managing decentralized ledger database in oil and gas industry,’’ MDPI J. Multidisciplinary, vol. 2, no. 3, pp. 300–325, Aug. 2019.

[72] R. Zhang, R. Xue and L. Liu, "Security and privacy on blockchain", *CoRR*, vol. abs/1903.07602, 2019.

[74] https://www.investopedia.com/terms/t/target-hash.asp. accessed 03/02/2021

[75] Aljabr, Ahmad & Sharma, Avinash & Kumar, Kailash. (2019). Mining Process in Cryptocurrency Using Blockchain Technology: Bitcoin as a Case Study. Journal of Computational and Theoretical Nanoscience. 16. 4293-4298. 10.1166/jctn.2019.8515.

[76] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei and C. Qijun, "A review on consensus algorithm of blockchain," 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 2017, pp. 2567-2572, doi: 10.1109/SMC.2017.8123011.

[77] N. Alzahrani and N. Bulusu, "A new product anti-counterfeiting blockchain using a truly decentralized dynamic consensus protocol", *Concurrency Computation: Practice Exp.*, vol. 32, no. 12, pp. e5232, 2019.

[78] King, S., Nadal, S.: PPCoin: peer-to-peer crypto-currency with proof-of-stake (2012). https://archive.org/details/PPCoinPaper

[79] https://www.moneyland.ch/en/proof-of-importance-definition. Accessed 3/2/2021

[80] Miguel Castro and Barbara Liskov. 1999. Practical Byzantine fault tolerance. In Proceedings of the third symposium on Operating systems design and implementation (OSDI '99). USENIX Association, USA, 173–186.