AFRL-RI-RS-TR-2020-107

DATA PROVENANCE ASSURANCE IN CLOUD USING BLOCKCHAIN

OLD DOMINION UNIVERSITY

JUNE 2020

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2020-107 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
LAURENT Y. NJILLA
Work Unit Manager

/ S /
JAMES S. PERRETTA
Deputy Chief, Information
Exploitation & Operations Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JUNE 2020	FINAL TECHNICAL REPORT	AUG 2018 – DEC 2019

4. TITLE AND SUBTITLE

Data Provenance Assurance in Cloud using Blockchain

5a. CONTRACT NUMBER
FA8750-18-1-0075

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)

Sachin Shetty

5d. PROJECT NUMBER
BLOC

5e. TASK NUMBER
KC

5f. WORK UNIT NUMBER
02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Old dominion University
Research Foundation
4111 Monarch Way, Suite 204
Norfolk VA 23508

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIGA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2020-107

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The project started on Apr 06, 2018. In the 24 months of executing the project, the team has conducted basic research on data provenance architecture for cloud using block chain, surveyed the vulnerabilities in block chain, in-depth analysis of the block discarding attack, developed a Proof-of-Stake consensus protocol for cloud based blockchain, architecture for secure BattlefieldIoT, cyber supply chain provenance, integration of software guard extensions on distributed ledgers for increased privacy.

15. SUBJECT TERMS

Blockchain, cloud, data provenance, consensus protocol, directed acyclic graph, fault tolerance, smart contract

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**LAURENT Y. NJILLA**
U	U	U	UU	99	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

Table of Contents

List of Figures

Figure **Page**

List of Tables

Table **Page**

1. EXECUTIVE SUMMARY

Emerging Internet of Things (IoT) technology is becoming a major part of our society to enhance operational efficiency of the existing infrastructure. Through advanced sensors and actuators, environmental data can be collected from various end-points and used to analyze for taking necessary control actions. In particular, the modern battlefields are equipped with advanced IoT-enabled weaponries, wearables, and vehicles, to increase the accuracy of decision taking capability during military missions. Although the operating devices in battlefield can have resource constraints such as storage, processing capability, and networking ability, security of data exchanged among these devices is critical to mission's success. The massive scale, heterogeneity, and distributed characteristics of Internet-of-Battlefield-Things (IoBT) present challenges in realizing a practical and effective security solution. Blockchain empowered platforms and technologies have been proposed to address such challenges by utilizing its tamper-resistant ledger. However, the amount of data generated in the IoBT network need to be validated and verified in real time in order to authenticate the mission-related data. Therefore, the core Blockchain infrastructure must be capable enough to meet these demands.

In this project, the key contributions are a) Blockchain simulator for IoBT environment, b) Lighweight Blockchain with sharding, c) Techniques for optimizing memory pools against flooding attacks and framework for characterizing blockchain based systems for accurate systems. The Blockchain simulator evaluates the consensus algorithms in a realistic and configurable network environment. Though, there are several Blockchain evaluation platforms, they are either wedded to a specific consensus protocol and do not allow evaluation in a configurable and realistic network environment. In our proposed simulator, we provide the ability to evaluate the impact of the consensus and network layer that will inform practitioners on the appropriate choice of consensus algorithms and the impact of network layer events in congested or contested scenarios in IoT. To accomplish this a generalized representation for consensus methods is proposed. The Blockchain simulator uses a discrete event simulation engine for fidelity and increased scalability. We evaluate the performance of the simulator by varying the number of peer nodes and number of messages required to find consensus.

FastChain is a simulator built in NS-3 which simulates the networked battlefield scenario with military applications, connecting tankers, soldiers, and drones to form Internet-of-Battlefield-Things (IoBT). Computing, storage, and communication resources in IoBT are limited during certain situations in IoBT. Under these circumstances, these resources should be carefully combined to handle the task to accomplish the mission. FastChain simulator uses Sharding approach to provide an efficient solution to combine resources of IoBT devices by identifying the correct and the best set of IoBT devices for a given scenario. Then, the set of IoBT devices for a given scenario collaborate together for sharding enabled Blockchain technology. Interested researchers, policy makers, and developers can download and use the FastChain simulator to design, develop and evaluate blockchain-enabled IoBT scenarios that help make robust and trustworthy informed decisions in mission-critical IoBT environment.

2. INTRODUCTION

Military environments are increasingly dependent on IoT-enabled devices to aid decision making. Despite resource constraints such as storage, processing capability, and networking ability, verifying authenticity of the devices and the integrity and provenance of the information exchanged by the devices is critical. The massive scale, heterogeneity, and distributed characteristics of Internet-of- Battlefield-Things (IoBT) present challenges in realizing a practical and effective security solution. We have proposed a Blockchain empowered platform for a resource constrained IoT environment [1]. Leveraging the tamper-resistant ledger can help achieving trust in IoBT, but it imposes various challenges in terms of network infrastructure, and real time quality of service requirements. The amount of data generated in the IoBT network must be *stored and need to be validated as well as verified in real time* in order to authenticate and validate the device operation.

Although the provably secure cryptographic primitives in Blockchain make it a suitable platform to avail security services for IoBT, the foundational layers of Blockchain platforms such as consensus and network layers are the main bottlenecks in achieving higher transactional throughput. Researchers and practitioners have pointed out the main performance benefits of Blockchain are closely tied to the choice of consensus plane and the network plane [2].

The consensus mechanism describes the process to confirm group of trustless transactions. However, there exist several consensus protocols, such as, Proof of Work, Proof of Stake, Delegated Proof of Stake, Practical Byzantine Fault Tolerant, etc. In order to realize a consensus model in resource-constrained and dynamic environments, such as IoBT, a single consensus mechanism may not be sufficient. There is a need for a reconfigurable consensus plane that will facilitate the ability to deploy hybrid consensus protocols in an on-demand basis. The flexible network layer is also responsible for ensuring that the transaction validation messages will reach to the peers in a bounded time and the network will be resilient to failures. Most Blockchain environments offer a static network topology and do not provide the ability to configure the network plane, as needed, to evaluate the feasibility in resource constrained environments, such as, IoBT. Thus, there is a need to develop a Blockchain environment comprising of a configurable network and consensus plane that can provide a trusted framework for IoBT environments.

In this projecte, we have developed a simulator that would provide the ability to analyze and to validate the claimed performance advantages of a practical Blockchain platform prior to deploying the technology in a production environment. Understanding that the IoBT network may have wide variety of devices, different network topologies, and bandwidths, it is of utmost important to address the scalability issue using a *graph-theoretic model* where the underlying network topology including all sensors can be interpreted as a graph. To improve the reliability of IoBT networks, and to reduce/discard the invalid data transactions/information, it is important to devise a mechanism that will relax the underlying network topology in such a way that the resourceful nodes will be fairly used while removing specific edges and limiting the sensing rates.

3. METHODS, ASSUMPTIONS AND PROCEDURES

The impact of the vagaries in the network layer on the Blockchain performance due to *node mobility* and intermittent *connectivity*, in IoBT environments, has motivated us to develop a Blockchain simulator. We propose the development of a Blockchain simulator that would provide the ability to evaluate the performance of consensus protocols in diverse and dynamic IoBT environments under various networking conditions. The simulation would provide users the ability to swap in and out various consensus protocols, configure the network settings, and assess the performance of the consensus protocols. Next, for several IoBT networking configurations, we will provide the ability to develop a *meta-consensus mechanism* that would allow one to integrate multiple consensus protocols based on the QoS requirements.

3.1 Generalized Consensus Blockchain Simulation Platform

The massive scale, heterogeneity and distributed nature of Internet-of-Things (IoT) presents challenges in realizing a practical and effective security solution. Blockchain empowered platforms and technologies have been proposed to address aspects of this challenge. In order to realize a practical Blockchain deployment for IoT, there is a need for a testing and evaluation platform to evaluate performance and security of Blockchain applications and systems. In this project, we present a Blockchain simulator that evaluates the consensus algorithms in a realistic and configurable network environment. Though, there are several Blockchain evaluation platforms, they are either wedded to a specific consensus protocol and do not allow evaluation in a configurable and realistic network environment. In our proposed simulator, we provide the ability to evaluate the impact of the consensus and network layer that will inform practitioners on the appropriate choice of consensus algorithms and the impact of network layer events in congested or contested scenarios in IoT. To accomplish this a generalized representation for consensus methods is proposed. The Blockchain simulator uses a discrete event simulation engine for fidelity and increased scalability. We evaluate the performance of the simulator by varying the number of peer nodes and number of messages required to find consensus.

3.1.1 Methodology

A set of abstraction layers, such as (1) Network; (2) Consensus; (3) Storage; and (4) View and Side, have been proposed [1] in past to provide a hierarchical design structure for the decentralized Blockchain ecosystem. Testing performance of specific consensus based algorithms is studied in the literature [2-3], but a general model for network layer evaluation of Blockchains is yet to be proposed. It is important to understand the complexity and performance metrics at each layer prior to deployment of Blockchain-based solutions. Therefore, we propose the development of a simulator that implements these layers and provides statistical insights on performance and security metrics via realization of the Network and Consensus layer.

Our choice of focusing on consensus and network layer is driven by the need and potential of Blockchain deployment in IoT environments where nodes are heterogeneous and have stringent constraints on networking, computation, and communication capabilities. Given the bottom-up design [113], the consensus and network layer is the core of Blockchain platforms and its

cost/complexity needs must be analyzed prior to integration of the additional layers. This approach will provide inner insights for the community of Blockchain adopters about the performance and security metrics of the underlying consensus protocols under various network conditions. In Figure 1, the architecture overview is presented which incorporates different network topology for establishment of a P2P network and implements various distributed consensus protocols for evaluation of performance metrics. We present an overview of the proposed architecture wherein consensus protocols and the network architectures are interchangeable.

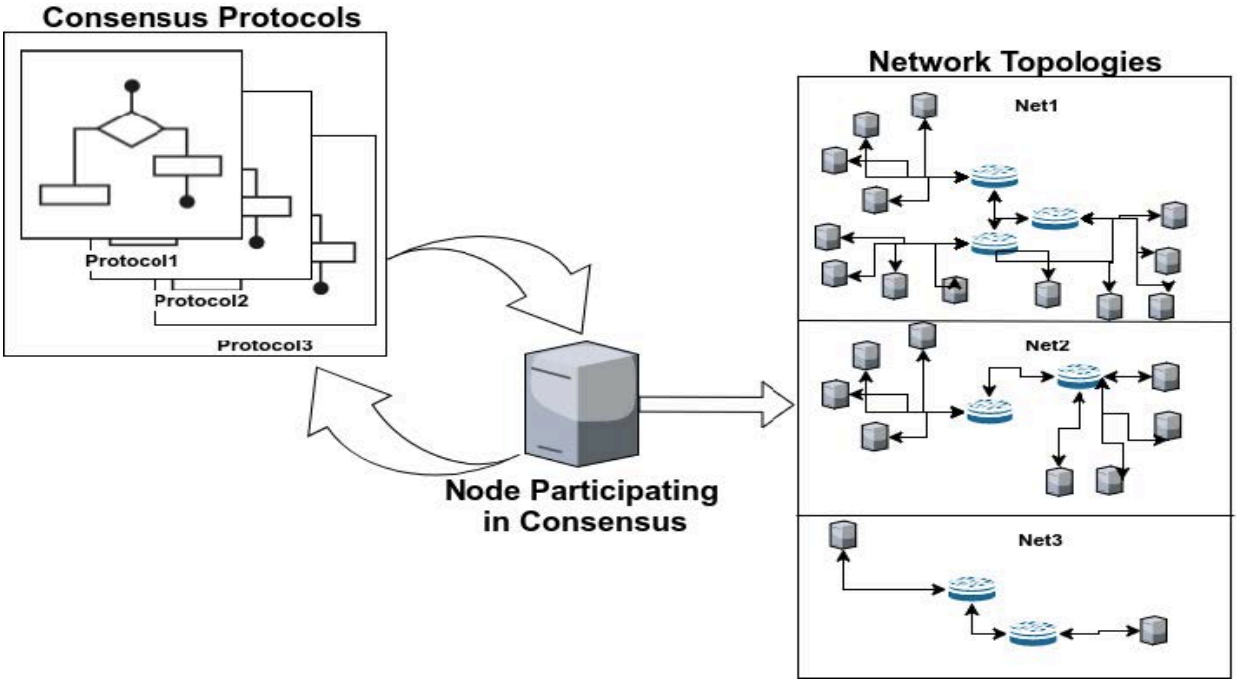


Figure 1: Consensus and Network Layers in Blockchain Simulator

3.1.2 Establishing Discrete Event Simulator

We propose a discrete event simulator to implement the consensus and network layers wherein the simulation time step is considered to be system events rather than regular time steps to facilitate quicker computation. The design goals considered while developing the simulator are:

1. consensus protocols need to be generalized,
2. underlying network needs to be re-configurable,
3. simulator must be able to handle large networks so that tests of scale can be performed.

The interchangeability of consensus protocols will ensure that the key aspect to the Blockchain systems performance is considered and evaluated instead of efficiency of the whole platform itself. The simulator will contribute to the benchmarking of consensus protocols for different network topologies. Using the simulator, developers will be able to make informed decisions on which consensus protocol would be preferred in certain types of network environments where

the Blockchain is considered for by understanding the performance metrics of each protocol, which is discussed in the following subsection.

A) **Performance Metrics**

The key metrics considered in the simulator are discussed briefly in the following.

- *Throughput:* Number of successful transactions/second.
- *Latency:* Response time/transaction.
- *Fault Tolerance:* Ability to find consensus and complete transactions with sub-optimal network topology.
- *Heterogeneity:* Meet above three requirements under heterogeneous network conditions and device characteristics.

The performance and resilience evaluations of consensus protocols on the considered Blockchain-integrated network are mainly conducted at the node level. Modeling the system at the node level enables better representation of the specific protocols' reliance on number of nodes and how they operate with varying network topologies. Therefore, we first observe and record the key aspects from an ideal system that has minimum influence from external factors, which is discussed more in the following subsection. Then, we use the observed data to model various components of the simulator for performance evaluation. Figure 2 shows the collection and use of the data from both the ideal system and the simulated system where blue indicating data gathered from individual nodes and red indicating data gathered from total system. Finally, we fine tune the developed model and validate to provide a meaningful measure on its effectiveness to represent the ideal system.

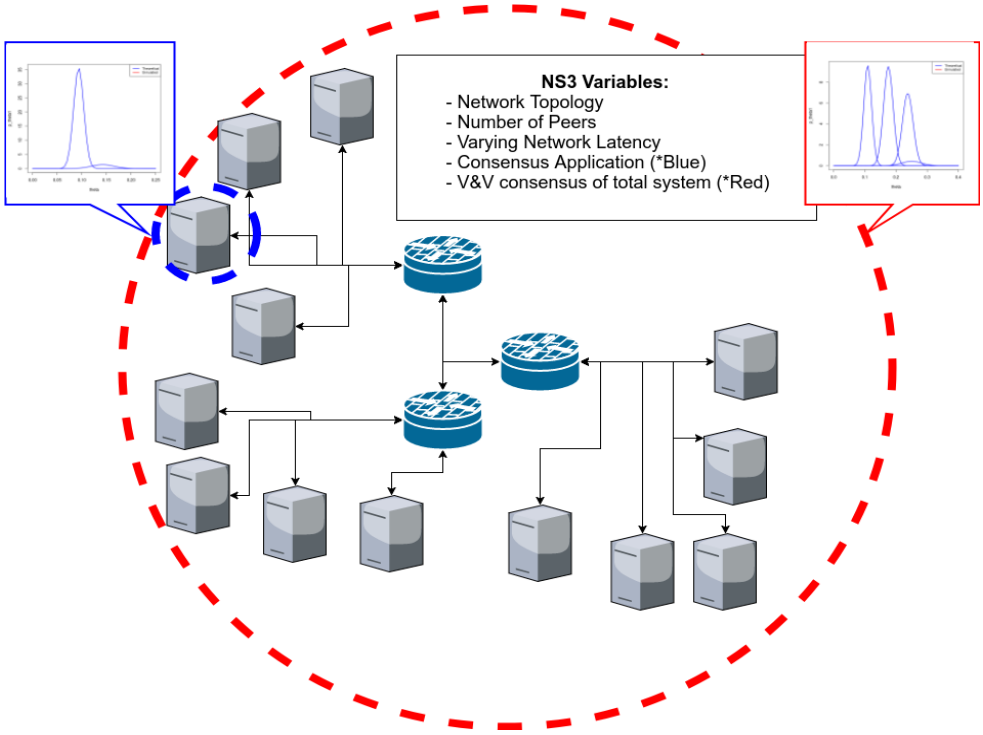


Figure 2: Demonstration of the data collection from the ideal system

B) **Ideal System**

An ideal system is a set up of the consensus protocols where the external engagements and influences are minimum. The purpose of the ideal system is to provide a platform that can be easily modeled which will provide results for the baseline scenarios. In this context, the ideal system has several nodes where every node may have either transient or permanent network connectivity with each other in a mesh topology. A good candidate for implementing this is to use a virtual network and containerized technology on a single machine using docker containers [4]. The docker nodes in an ideal system act as validator nodes and run the consensus protocol. The number of networked nodes is kept variable to identify how performance metrics change as number of nodes scale. Logging is also enabled on the individual nodes to record performance aspects of the consensus protocol. In the next subsection, we discuss different kinds of gathered data and collection mechanism to better understand the state and performance of the system.

C) **Data Collection**

The ideal system provides the ground truth for the proposed simulator and will be run with each consensus protocol modeled. We collect various features in order to represent the key performance metrics of the modeled system as discussed earlier. The details of data points to be recorded are discussed below.

- *nMessage:* Numer of exchanged messages to achieve consensus
- *nodeDelay:* Processing delay at node level required for achieving consensus.
- *nodeConsen:* Number of participating nodes required to achieve consensus
- *consenDelay:* Network delay for the consensus protocol to achieve consensus
- *consenPauseState:* Probability that the consensus state is paused for a special case
- *pauseStateTime:* Time that the consensus system remains in the paused state

D) **nMessage**

In the ideal system from the participating node the number messages will be recorded during the consensus process. The data *nMessage* will be recorded by counting the number of messages received and sent while consensus is being determined. The flow for recording this is as followed:

- A transaction is proposed to the consensus system
- from the participating node increment received counter when messages received
- increment sent counter responding messages sent
- when consensus is found report counter results and initialize them to zero

This process will be repeated several times in order to get a good representation of the number of messages a participating node must process during the process of finding consensus. The total data set can then be processed and fitted to a distribution that represents the number of messages a participating node processes during consensus

E) **nodeDelay**

The *nodeDelay* can be recorded the same time the *nMessage* is recorded. The participating nodes

take a small amount of time to process each message during the consensus process. The data *nodeDelay* is essentially the time delay between each message received and each message sent during the consensus process. The flow for recording is provided below:

- A transaction is proposed to the consensus system
- from the participating node when a message is received record a time stamp
- when the participating node responds record a time stamp
- initialize both time stamps
- report received and sent timestamps
- when consensus is found output time stamps for processing

Data will be gathered for a number of messages preferably simultaneously to the recording of *nMessage*. Once the data has been collected, the difference between the timestamps of the messages can be processed to derive the time it took the participating node to process each message. This data can then be used to fit a distribution that represents the typical time to process messages during consensus.

F) **nodeConsen**

The *nodeConsen* can be obtained from literature or documentation for the consensus protocol. The purpose of *nodeConsen* is to find a threshold value for how many nodes need to functioning properly for the system to operate. This is often referred to as the byzantine node threshold or the number of nodes required for the system to function properly. If this does not exist, the measure can be obtained by running the ideal system and then removing nodes until the system no longer finds consensus or the consensus is incorrect. This measure can be a static number or a function based on the number of participating nodes.

G) **consenDelay**

The *consenDelay* can be recorded from the ideal system while the other measurements are be recorded. It is the time it takes the consensus system as a whole to find consensus. This time can be realized by the start time of when a transaction is requested and the time that consensus is found by the system. The flow for recording this is as followed:

- A transaction is proposed to the consensus system
- a time stamp is recorded at the time of the proposal
- when the system finds consensus a time stamp is recorded
- time stamps are reported and all time stamp variables are initialized

Data will be obtained for many transactions to get a large sample of consensus delay time stamps. These time stamps can be processed after recording to identify the difference between the two time stamps resulting in the time that each transaction took to find consensus. This data is then divided in two sets one set for calibration of the model and simulation during development and another set for the validation of the model and simulation.

H) **consenDelay**

Certain consensus protocols have periods of restructuring or organizing that occur when nodes go down or at various periods of time during operation. This special state of the system could be

a leader selection process or anchor node assignment. This process is period where the system may delay transaction until the state is resolved based on each protocols own process. Once this state is resolved, the consensus protocol goes back to its normal transaction state. The measure *consenDelay* is the probability of at this moment in the simulation that the consensus system will go into this special state halting transactions until resolved. To obtain this value, literature of the consensus protocol can be utilized to identify if there is a periodic time when a special case occurs. This might happen when a change to the participating nodes occurs. If this is the case the *consenDelay* can be assigned to 1 when the network topology changes for the participating nodes.

I) **pauseStateTime**

For the special states, the time to process it needs to be identified. The variable *pauseStateTime* will hold this value for each consensus protocol. This measure is the amount of time the consensus protocol takes to typically resolve the special state and continue processing transactions again. This measure will be obtained by identifying when the ideal system consensus protocol enters this special state and exits the special state. This measure will be sampled many times so that the data set can be fitted to a distribution to represent the possible values that represent the special state length.

J) **Consensus Layer Modeling**

To fully represent the consensus protocols' effects on different numbers of participating nodes, limitations on network, computing resources, and network topology, several simulation runs with samples of the input data is needed. Hence, to complete these runs in a reasonable amount of time, discrete event simulation is used, where a networked message is considered as an event. The discrete event simulation engine schedules each message as the simulation runs based on how each modeled component affects the system. The component affects are modeled as statistical representations of observed data from the ideal system or distributions.

Each participating node runs a validation instance that models the actual workload of a peer in the consensus protocol. The validation process includes the necessary inputs required to represent the underlying consensus protocol steps. All the input data for each state of the consensus protocol is stored in the form of distributed libraries or variables in the application. As the simulation runs, the participating peer either begins a new transaction or participates in the consensus of an existing transaction. The participating peer continues this operation until either a certain number of transactions have been processed or a period of time has elapsed.

Transaction processing starts when one of the current peers proposes a transaction to the system. It causes initialization of the transaction module, which then sends out messages to the participating peer nodes. The transacting node schedules the message to be delivered to every peer node that it can communicate with on the simulated network through utilizing the existing routing libraries. Based on the simulated arrival time of the message to each node, the simulation engine orders the events accordingly where the events are handled in the order they are scheduled in the event list.

As participating peer nodes receive messages from the consensus process, they internally record

the number of messages processed during the current consensus iteration. If a threshold number messages have been processed by a peer node as determined by *nMessage*, then the node's state is assigned to ``completed". Peer nodes with status ``completed" send a ``completed" message to the other participating nodes stating that it has completed its consensus round. If the peer node has not processed enough messages, it will schedule another process message with the simulation engine for a delay sampled value based on the *nodeDelay* distribution. The node continues to process messages until the appropriate number of messages have been processed and upon completion, its state changes to ``completed". The leader peer that started the transaction monitors the system until *nodeConsen* number of participating nodes reach a state of ``completed". After enough participating nodes have reached a ``completed" state, another random peer can start a transaction.

The general consensus system is modeled to change leaders periodically based on the consensus protocol it is modeling. If the system changes states, the transactions will be halted for period of time *pauseStateTime*). The next event would be scheduled with a new transaction and the start time is set after the delay *pauseStateTime* completes.

3.1.3 Implementation

In this Section, we discuss the steps taken to implement the proposed Blockchain simulator in details. The simulation paradigm is considered to be discrete event simulation. The general consensus protocol described in the methodology is built on Network Simulator-3 (NS3).

A) Simulation Data Gathering

For this initial test, Raft [5] is chosen as the consensus protocol because of its simplicity and the many available packages that exist to execute just the consensus protocol without any additional features. Running full packages of systems that utilize consensus protocols could produce results that are inflated based on the additional processing required to run additional features, hence for this paper just the consensus protocol is of interest. Raft is set up to run on a single machine with three to five participating nodes. For each of these settings one of the nodes is the leader node the entire time. The nodes run as docker containers on a single machine using a virtual network that provides the fastest connection possible. This environment ensures that the time to process each message is obtained without the effects of network delay. The time to find consensus is recorded for one thousand iterations and the frequency plot is shown in Figure 3.

The time for each node to process a single message of those thousand iterations is shown in Fig 3. We set up Raft to run on a single machine with three to five participating nodes. For each of these settings one of the nodes acts as a leader node throughout the test. The nodes run as docker containers on that machine and are connected through a virtual network to provide as ideal connectivity as possible so that the time to process each message does not get effected by network delays.

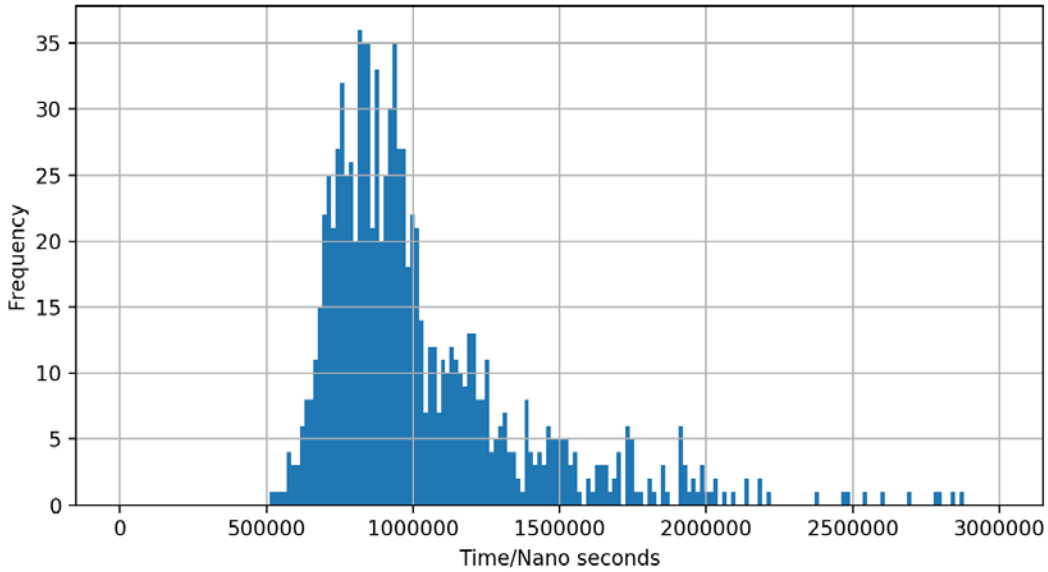


Figure 3: Histogram of time for Raft to find consensus

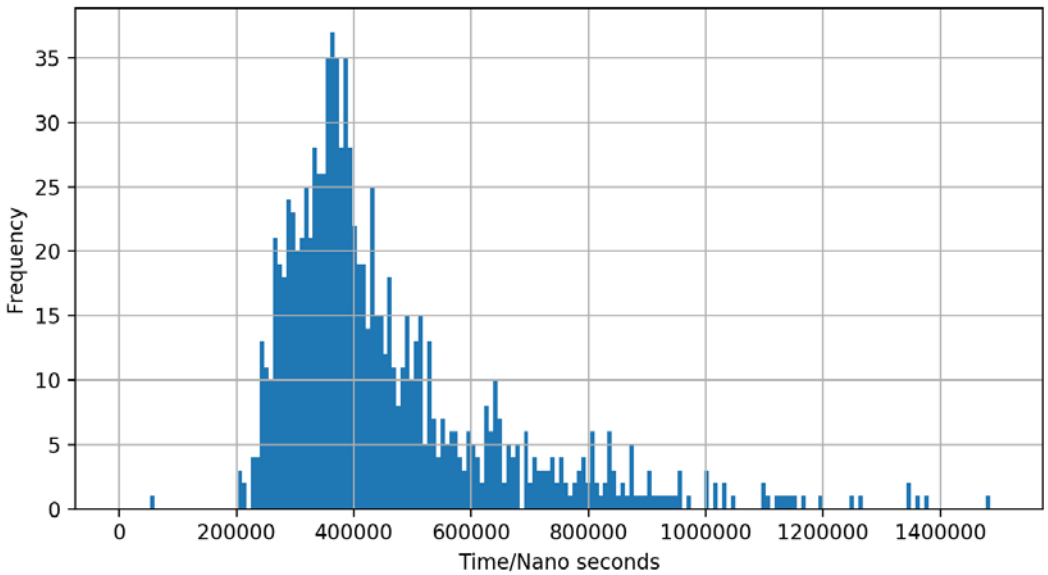


Figure 4: Histogram of time for a peer node to process a Raft message

As discussed, consensus data is collected for the Raft protocol to test the methodology. We record the time to find consensus for one thousand iterations and the time for each node to process a single message in each of the thousand iterations. We subsequently fit the gathered data to a distribution. Figure 5 shows the distribution of a node's processing data represented as a histogram and the result of the curve fitting (using R stats) for Weibull, Lognormal, and Normal

distributions. It is observed that node's processing data for the Raft protocol follows the Weibull distribution as it best fits the ideal Raft consensus with a scale parameter of 1.35 and a shape parameter of 249857.5.

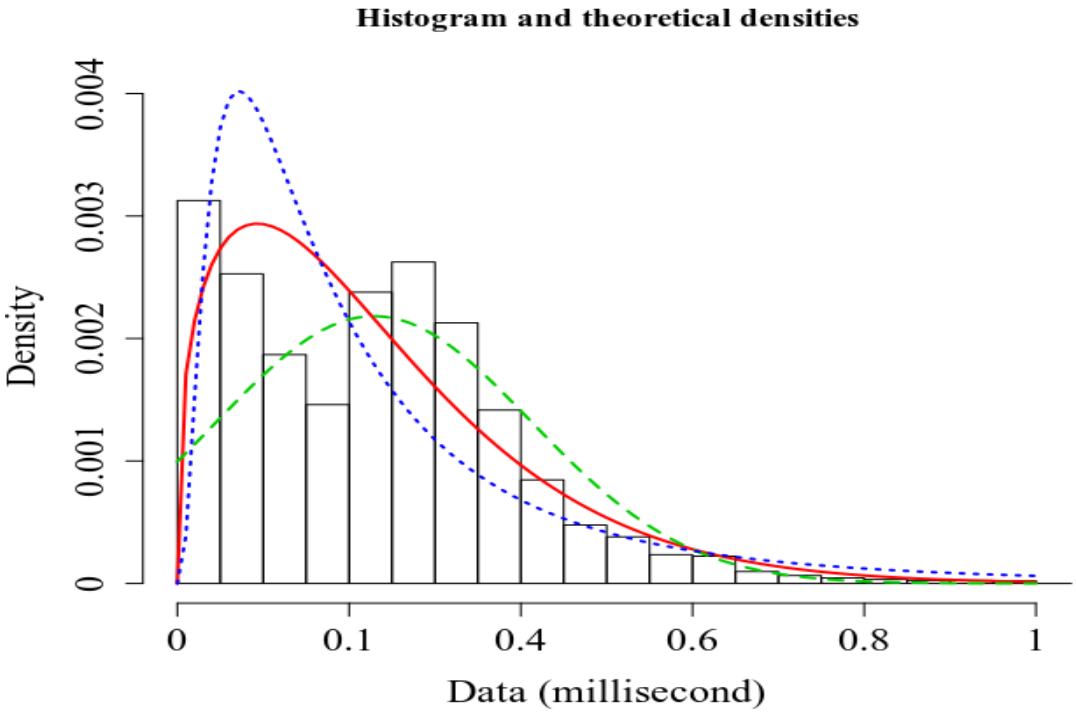


Figure 5: Curve fitting node message process times of Raft peers

B) Network Layer Implementation

The network simulator handles the flow messages among the participating peers. Researchers have implemented a proof-of-work consensus protocol with the Bitcoin-Simulator using NS3 [115]. It has the ability to generate large scale networks and communicate using various networking methods such as P2P and gossip-relay types of network. For this paper, we modified the main controller module, as well as the Bitcoin node and built a new application extending the Bitcoin node for the general consensus node.

The network layer is built to represent the ideal system that is used to gather data for the simulation. To best represent this, the simulator uses a mesh P2P network using the point-to-point NS3 module. The links use data rate in the order of gigabits per second, and the link delay is set to zero with the intention of modifying this for calibration. Although the Bitcoin simulator is capable to generate more complex network topologies spanning multiple regions of the world, in this paper we consider that all nodes are connected through a mesh P2P network in the same region.

C) Consensus Layer Implementation

As described in the methodology section, each participating node has certain variables and distributions to represent the behavior of the consensus protocol it is modeling. A leader node is specified at the beginning and will broadcast a message to the participating peers indicating the start of consensus. The message is sent using the NS3 socket class and the send ability. NS3 handles the message as a set of discrete events traversing the network. The peer nodes then proceed to process consensus messages as specified in the methodology section and consensus is found when *nodeCompleted* nodes have processed *nMessages*.

D) Calibration and Simulation Sensitivity

With the model developed and functioning based on the observed data from the participating nodes, the total system is calibrated to observe how close it performs with respect to the ideal system. Calibration is used to tune the model so that it performs good enough compared to the ideal system. Tuning the model and simulation is done by increasing network background traffic or adding delay to network to create more delay in the message processing. Calibration is done by tuning the distributions used to represent the message processing. This is done using constants as a scaling factor for selected input data sets as needed. In our proposed simulator, calibration is done by:

a. increasing network background traffic or adding delay to network to create more delay in the message processing,
b. tuning the distributions using constants as a scaling factor for selected input datasets as needed.

It is an iterative process and can use optimization methods to minimize the error in time to find consensus between the model system and the observed data from the ideal system.

3.1.4 Results and Discussion

In this section, we provide the results obtained. First, we show the observation of the data gathered and how we fit it in the curve. Then, we show the calibration results with the sensitivity observations. Based on these observations, we use a curve fitting software to find the best distribution to represent the time a node takes to process a message. The MASS library for R stats is used for the curve fitting and Figure 5 shows the results of the node processing data represented as a histogram and the results of the curve fitting software for weibull, lognormal, and normal distributions. For the node process the weibull distribution is the best fit for representing the raft consensus.

A) Network Layer Observations

The network delay is modeled using a lognormal distribution. The lognormal mean and variance are found to be good at 12.85 and 0.57 respectively. These values are fitted with an iterative process, running the simulation comparing results, then we use these results to adjust the parameters and feedback this change to the model. This process continues for several iterations until a close fit is found. The calibration is done for the ideal system, with three peers running the Raft implementation. %which utilized one message from each peer.

The results of the calibrated model are shown in Figure 6 with two scatter plots. The scatter plot on the left is calibration of the simulation to the ideal system. The x-axis represents time in seconds for a consensus transaction by the simulator. The y-axis represents time in seconds for a consensus transaction by the ideal system. The red straight line is the computed regression line of the two data sets. The left scatter plot shows that there is a tight grouping and positive correlation with the simulation and the observed ideal system. The scatter plot also shows that for the few extreme values the two systems differ some. This is likely because the curve fitting is mostly matching the common results in the observed data. The distribution could be modified and calibration could be pursued further to better match the extreme values.

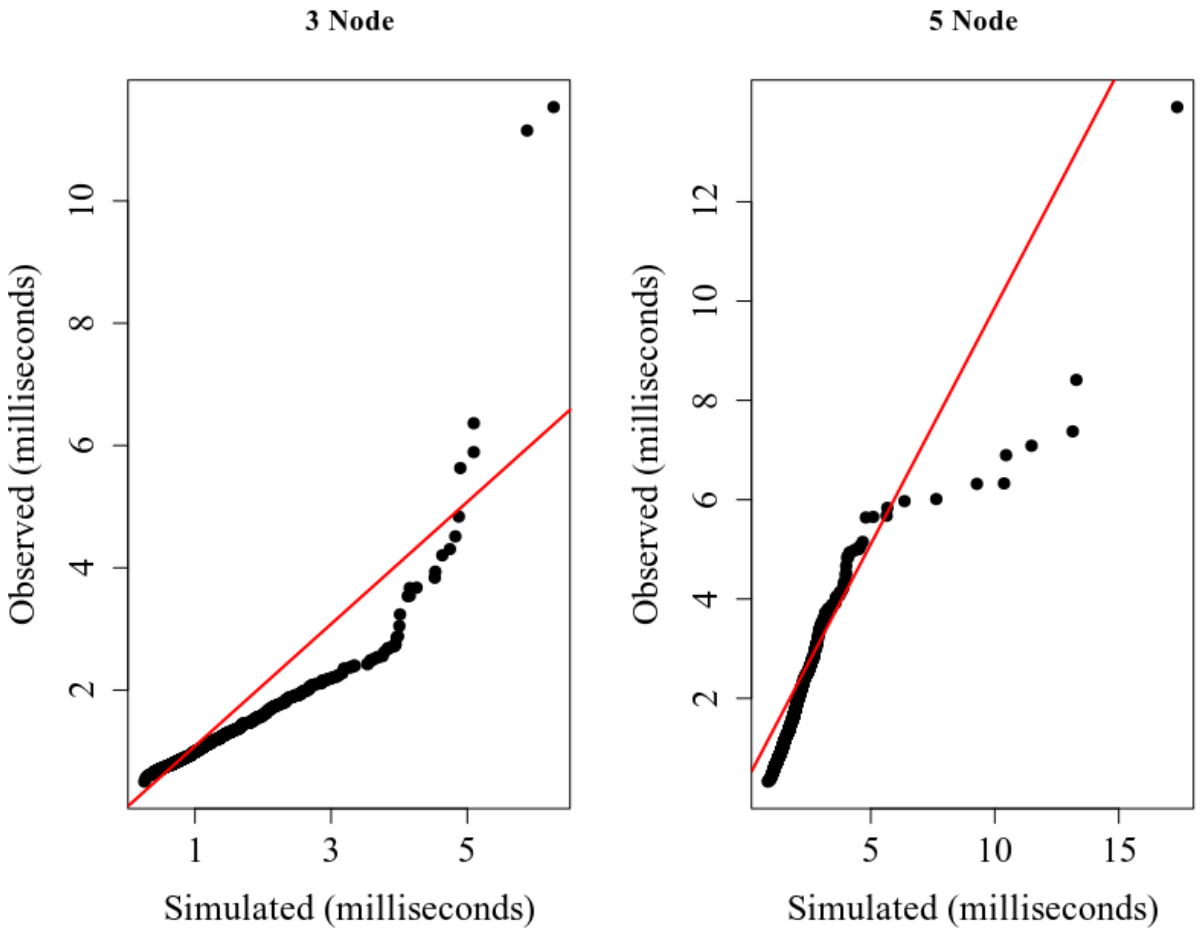


Figure 6: Calibrating scatter plots of simulation versus observed consensus times

The scatter plot to the right in Figure 66 is the results of the calibrated model and simulation of the ideal system but run with 5-nodes instead of 3. The simulated results are compared with observed results for the same system. In this plot, the simulator is not calibrated for 5-peers, rather we wanted to observe if the behavior is good enough to represent the system's state using the calibration from the 3-peers scenario. This comparison in Figure 66 was done as a verification process to see if the behavior of the calibrated simulation is transferable to other scenarios.

B) Consensus Layer Observations

The calibrated model and simulation now allow us change the inputs to explore how the systems performance might change. At this point the number of required participating nodes can be changed to see how that will change the performance of the consensus system. The number of messages can also be altered to show how a different consensus protocol similar to rafts message processing might perform.

A sensitivity analysis is done where the number of peers and the number of messages required for the general consensus protocol are modified. This type of exercise gives insight into how much the simulation of the general consensus protocol changes and gives an idea of which input between the number of peers and number of messages has a greater effect on the resulting time to find consensus.

In order to get insight into how much the simulation of the general consensus protocol changes and understand the effect of the number of peers and the number of messages on the time to find consensus consensus, we conduct sensitivity analysis. In this analysis, we modify the number of peers and the number of messages required for the general consensus protocol, and observe their impact.

For this test, 30 simulation runs are conducted for each scenario to handle the stochastic nature of the sampled distributions used. The first scenario consisted of three required participating peers and the requirement for each one to process one message. The simulation processed 1000 consensus transactions and then recorded the time it took to complete the transactions. Table Consensus times shows the results for the test while varying the number of peers. In this table, the average time to process 1000 consensus transactions is shown for each number of peers tested. The data shows an increasing amount of time as the number of peers increases. The chart also includes error bars that provide information on the maximum and minimum values of the 30 simulation runs for each average data. This gives an idea on the amount of variability the simulator is producing in its results.

We can observe that the time to reach consensus increases as the number of peers increases. The table also includes the standard deviation and the minimum and maximum values as a percentage of the average to provide information on the variance over 30 simulation runs. This gives an idea on the amount of variability the simulator is producing in its results for each number of peers used. Generally speaking the standard deviation increases as the number of peers increases indicating more variance in the results. This is expected since the system becomes more complex with more participating peers.

Table 1: Consensus times (in ms) for 1-Message scenario

Peers	Average (ms)	Standard Deviation	Minimum	Maximum
3	1318.48	27.87	-3.96%	4.81%
4	1505.0	26.44	-2.69%	5.72%
5	1651.89	30.73	-3.68%	3.30%
10	2168.90	40.53	-3.96%	4.06%
16	2575.94	45.39	-4.01%	4.24%

In order to further observe the impact of the number of messages on the time to find consensus, we ran the above described simulation while increasing the number of messages. Figure 7 shows the same data for the 1 message test as shown in Table Consensus Times , but compared to similar tests with 4 message consensus and 10 message consensus. With this figure we observe increments in time to process 1000 consensus transactions between the different scenarios. Also, the time difference between different number of messages decreases when the number of peers increase. This finding provides some insight into the factors that cause the most delay. In this case it is shown that the number of peers has a greater effect on the delay of the system than the number of messages.

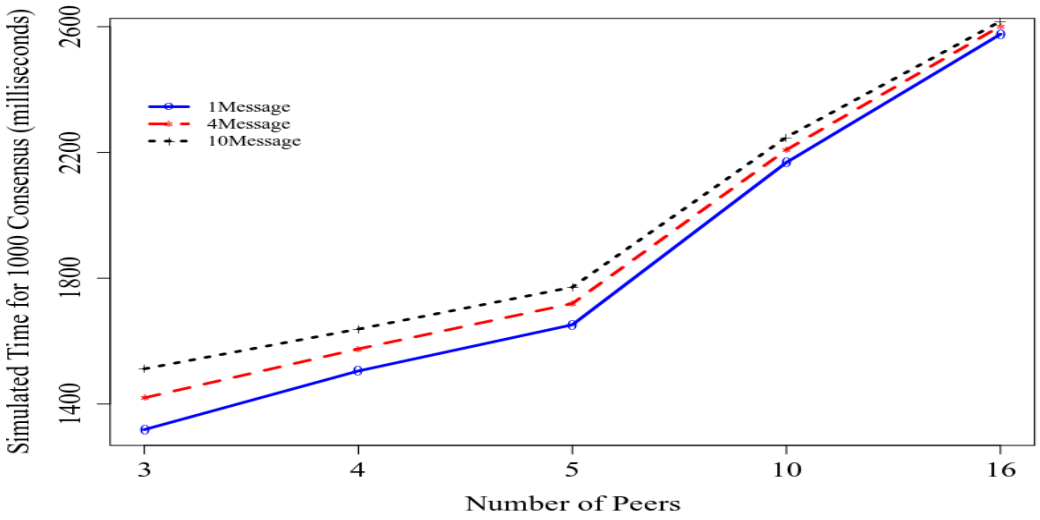


Figure 7: Average time (in ms) to confirm 1000 transactions

By observing the ability for the system to operate in various configurable network topology's we begin to study under what conditions the system fails to operate. This can occur when portions of the network do not allow the peer nodes to quickly message in participation of transactions. The unique characteristics of the simulator are a general consensus algorithm operating in a realistic and configurable network environment. The discrete event simulation engine allows to specify the consensus algorithm operations at faster than real-time fidelity without loss of scalability.

C) Network Topology Generation

To better test the effects of the network on the generalized consensus protocol a function was included to generate random network topologies. This function generates a simple representation of nodes and edges to form various topologies to test with. The random generation is uniform offering no bias in the process.

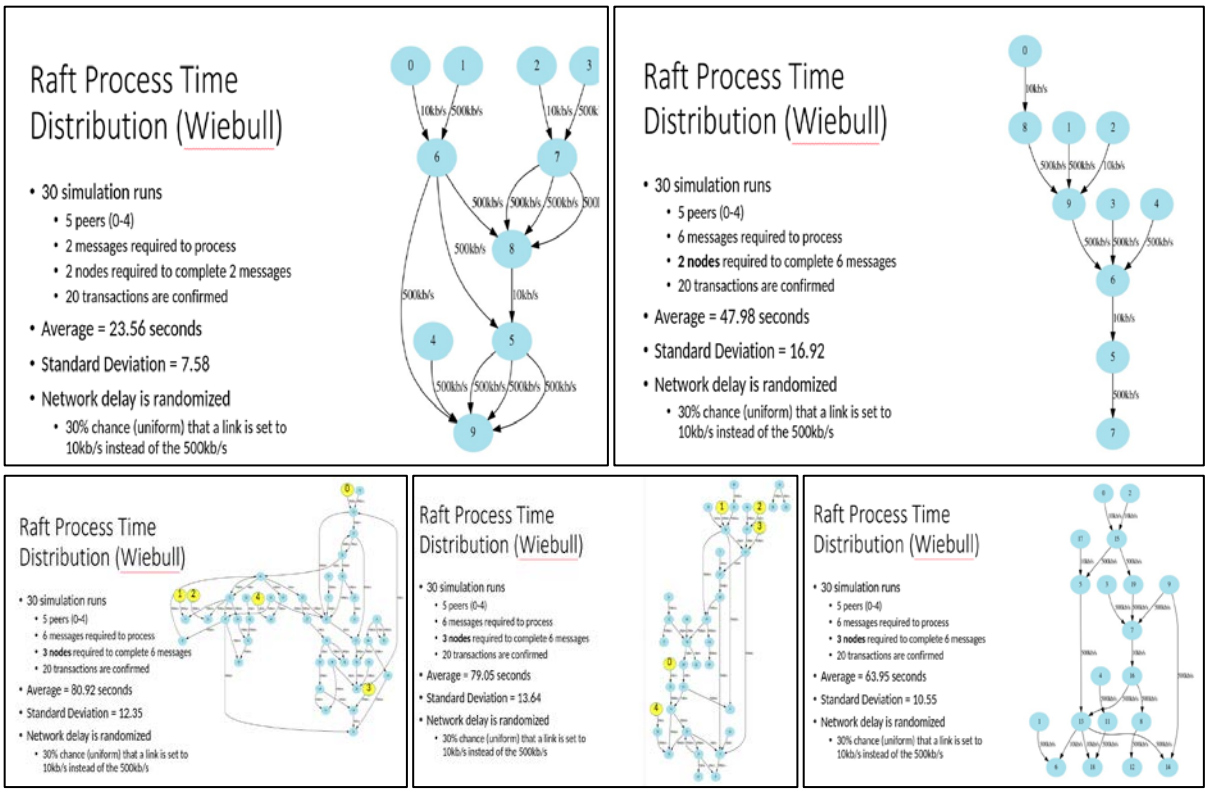


Figure 8: Randomly generated network topologies effects on the consensus protocol

The current functionality utilizes a peer to peer (p2p) network connection. The number of nodes is received as an input along with the number of peer nodes that will participate in the consensus protocol. The edges can assign a network speed throttling the speed of messages passed between the two nodes. All nodes are able to exchange information to and from any node they are connected to. For this simulation, all nodes that are not participating peer nodes do not exchange any additional messages, but only act as relay nodes for the active participating nodes. Future implementations can model background network traffic from all the nodes.

D) Project Integration with Simulation

Two projects are selected to integrate with the blockchain simulation. The titles of the two projects are:

1. PCBChain
2. Drone MPC

The two projects implement distributed systems on real hardware to indicate the proof of concepts for future systems. PCBChain is a proof of work variant able to run on small and low resourced devices. Drone MPC is an implementation of a Multi-Party Computation algorithm with commercial drone equipment through raspberry pi device communication over wifi.

The two projects will be integrated into the blockchain simulation using the general consensus protocol method. Parameters will be defined for both projects then implemented in the blockchain simulation. The simulator will be able to test these projects based on the reported

performance characteristics associated with their distributed system. With this ability, various network topology's can be tested with the system to offer insight in the effects different networks have on the performance of these systems.

E) PCBChain

This project implemented a lightweight consensus protocol for internet of things (IOT) devices that do not have large amounts of resource. These devices have limited memory or storage for large blockchain systems. This project focused on a proof of work (PoW) consensus protocol. The authors utilized a minimized hashing function to allow for faster hash generation in the PoW process. Extra hardware specific security was also implemented to aid in the performance of secure communication between participating peers that helps to mitigate device spoofing on the system.

The reported metrics included:

1. Verification of signers between nodes: 10ms
2. Hash rate (per second) 3.13mh/s
3. Transactions per second (TPS)
    a. One full node per lightweight node = 5tps
    b. One full node per 50 lightweight nodes = 350tps

The initial transaction values are determined based best case network topology's. This topology is represented as a centralized type of topology where each lightweight node I connected to the full node. The authors tested up to 6,250 lightweight nodes with either all of them working with the same full node or a full node assigned to up 50 light weight nodes.

F) Integration

The integration is ongoing but utilizes distributions of PoW success for varying difficulties. PoW utilizes text that is hashed. A participating node uses a nonce value at the end of the text and iteratively changes the nonce value until a hash value is found in which the difficulty is met. In the case of traditional PoW, the difficulty is enforced by requiring a certain number of zeros at the beginning of the hash value. A simple PoW implementation was used to sample the number of attempts that were used to find a successful hash value based on various levels of difficult.

In the case of PCBChain the 5tps at a hash rate of 3.13mh/s is roughly between a difficulty of 5 or 6. Using a simple PoW implementation attempts were recorded for thousands of blocks and fitted to a distribution. From the samples of different difficulty levels, exponential distribution was shown to be suitable. Figure 9 shows a histogram of the recorded data and the fitted data with lambda value of (9.899e-07)

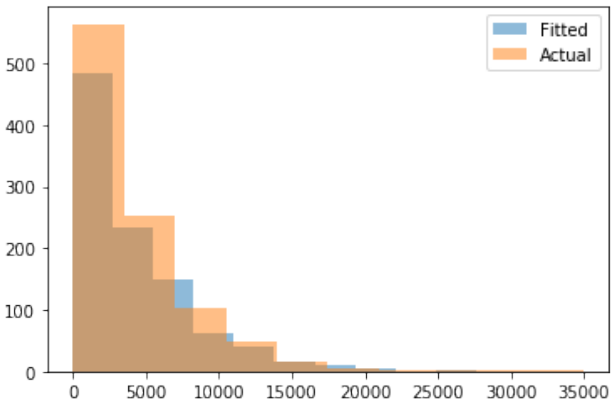


Figure 9: Exponential distribution fit of Proof of Work hash attempts

When the distribution is used with the hash rate of the PCBChain, the simulator gets a TPS value slightly higher than the expected 5tps of the observed system. A scaling rate of a ceiling rounded value of 23% is applied to the hashed results giving the final results a close match to the observed system.

G) Drone MPC

This implementation utilized raspberry pi comm link with commercial drone hardware. The Rasberry Pi participates as a peer node in a secure Multi-Party Computation based system. MPC algorithms are valuable in the use of obfuscating specific data while still being able to share aggregate data with the network. This system is distributed but not necessarily a consensus protocol, however, the system is distributed and relies on communication from all participating nodes in order to derive the correct aggregate information. The MPC algorithms can still be represented with the general consensus protocol using the number of messages required and the time to process a message.

The Drone MPC project success is partly from the use of its wifi communication between the Rasberry Pi and the participating Drones. There is a range limitation to this project and therefore can offer connectivity issues. The network simulator is capable of simulating varying frequencies of connection loss and timing of the connection loss. The network simulator will be adapted to handle the loss of connection between the drone and the Rasberry Pi unit. For this implementation an MPC application will be chosen that has metrics that are identified based on the number of messages and the ideal system computation time. One example of this is the following paper [6].

3.2 FastChain: Lightweight Blockchain with Sharding for Internet of Battlefield-Things in NS-3

FastChain is a simulator built in NS-3 which simulates the networked battlefield scenario with military applications, connecting tankers, soldiers, and drones to form Internet-of-Battlefield-Things (IoBT). Computing, storage, and communication resources in IoBT are limited during certain situations in IoBT. Under these circumstances, these resources should be carefully combined to handle the task to accomplish the mission. FastChain simulator

uses Sharding approach to provide an efficient solution to combine resources of IoBT devices by identifying the correct and the best set of IoBT devices for a given scenario. Then, the set of IoBT devices for a given scenario collaborate together for sharding enabled Blockchain technology. Interested researchers, policy makers, and developers can download and use the FastChain simulator to design, develop and evaluate blockchain-enabled IoBT scenarios that help make robust and trustworthy informed decisions in mission-critical IoBT environment.

3.2.1 Introduction

Internet-of-Battlefield-Things (IoBT) has been emerging as one of the major components of mission-critical military ap- plications where tankers, vehicles, war-fighters/soldiers and drones are connected and work collaboratively to accomplish the mission [7], [8]. However, the operations and commands coming from higher authority or peers could be compromised by the adversaries which could mislead the overall mission. Thus it is essential to have tamper-proof and trustworthy communications in IoBT. Distributed digital ledger also known as Blockchain is regarded as a next-generation consensus technology which does not depend on centralized trusted third-party [9], [10]. Blockchain is an open distributed ledger that can record transaction between two parties efficiently and in a verifiable and permanent way by definition [9], [11]. Blockchain is a peer-to-peer network which collectively coheres to a protocol for inter-node connection and validates and adds new blocks to the chain on a consensus basis. IoBT is the bare fruit of implementing BlockChain technology on the battlefield to be able to make trusted informed military decision using IoBT.

The overall intention of IoBT is to develop the fundamental dynamically-composable, adaptable and goal-driven networked battlefield environment to enable informed intelligent command and control operations to accomplish the assigned mission in the battlefield. In the near future, military operations will depend less on the human officials or warfighters and more on interconnected innovations and smart things/devices with advancements in embedded systems and machine intelligence to achieve superior defensive capabilities. The IoBT is expected to connect soldiers/war-fighters with advanced technologies to serve better in armor, radios, weapons, and other objects to grant troops with extra sensory perception, offer situational understanding, give better prediction power in certain situations, provide better risk assessment and develop shared intuition [12].

The huge scale and distributed nature of IoBT devices create many security and privacy challenges. The military mission scenarios constantly change and to adapt to that change, the underlying network and communication IoBT infrastructure needs to be flexible and adaptive. This can be achieved in an autonomous way, where no dependence on the centralized server should be sought. Moreover, the information propagating through IoBT devices should be verified to be from the trusted party and to be accurate. In addition, adversaries can interfere and compromise IoBT devices to impact the mission negatively. To address these challenges, enhance trust, and to enhance the credibility of the information in IoBT devices, emerging Blockchain technology should be utilized. Blockchain is an auditable platform to authenticate the accuracy of the information and create a tamper-resistant, robust and trusted environment for IoBT devices to communicate. Therefore, Blockchain is the perfect technology to integrate with network-centric mission-critical military operations. Furthermore, transparency and cost-effective properties of blockchain make it an appealing choice to be used in interconnected IoBT devices. A blockchain platform can have

many advantages to military cyber operations. The origin of every operation on cyberspace is recorded and traceable which ensures transparency. Once a transaction occurs, it cannot be reversed. The created transaction cannot be altered but a new amended one could be appended to the chain. The transactions in Blockchain are verifiable and can be audited as the ledger is tamper-resistant. Blockchain helps to establish decentralized trust among the entities and avoids depending heavily on one single entity.

This project report presents design, development, and evaluation of IoBT scenarios by marring the blockchain technology with IoBT devices [13], [14] using NS-3 [15]. IoBT is a complex network which needs not only highly scalable but also an accurate simulator to evaluate the complex IoBT. Simulation plays a vital role on IoBT research for experimental speed, scalability, reproducibility, experimenting the war-zone scenarios and training the war-fighters. Simulations can help study on small scale, large scale, simple scenario and complex scenarios to analyze the results and consequences without actual cost of creating real-battlefield testbeds. When IoBT relies on computing and storage resources of the IoBT devices, resources that are available on the battlefield might not have been utilized efficiently to achieve the goal of the mission. Simulators can help us how overall efficiency can be enhanced by effectively utilizing IoBT resources. This report presents, how FastChain simulator can help to simulate IoBT scenarios to group resources available in the Battlefield to implement Blockchain for trustworthy and temper-proof information and communications which help complete the mission in the battlefield. This grouping can be done depending on the computing, storage and transmission capabilities of individual IoBT nodes needed for implementing Blockchain technology in the battlefield. Then, groups in IoBT setup implement sharding enabled Blockchain [13], [16] for trustworthy and tamper-proof information and communications while establishing consensus needed for each IoBT devices. Specifically, we break down the approach taken to establish the IoBT context, create IoBT nodes, establish communications and form clusters by grouping IoBT nodes considering their computing, storage and transmission capabilities. The FastChain simulator helps visualize IoBT nodes representing the battlefield scenario and how resource limited IoBT devices can be grouped to implement sharding enabled blockchain technology [16].

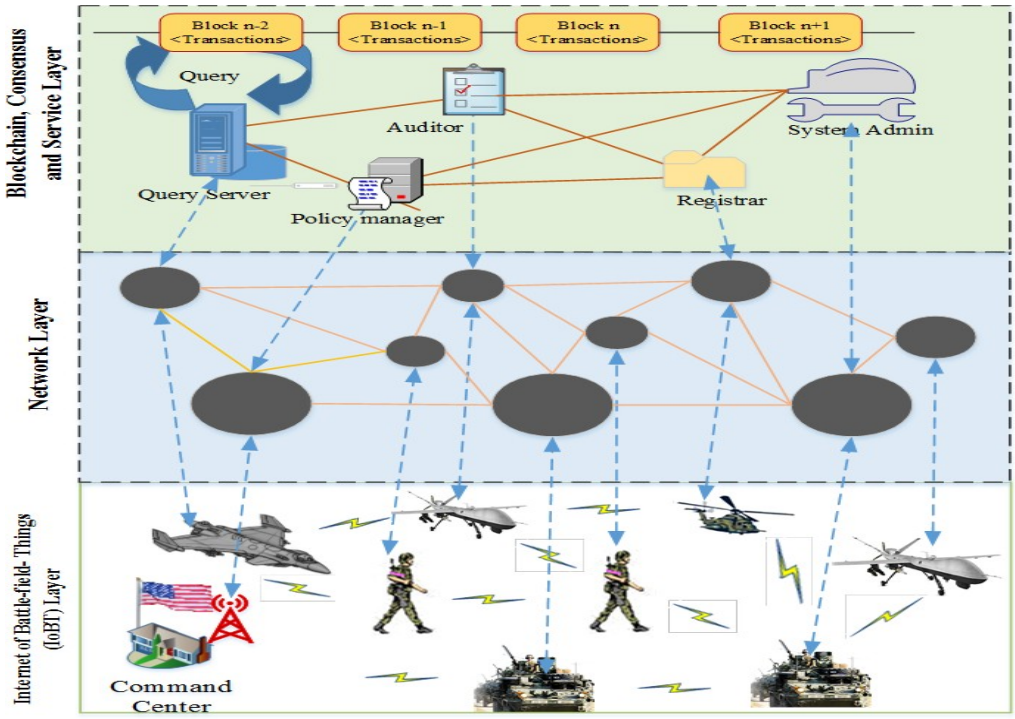


Figure 10: A typical block diagram of the Internet of Battlefield Things (IoBT)

3.2.2 System Model

A typical system model for IoBT consists of war-fighters, drones, tankers, and command centers as shown in the lower part of the diagram in Fig. 10. War-fighters, drones, and tankers deployed to accomplish the assigned mission through the command and control operations coming from command centers as well as based on the learned/observed surroundings of the IoBT nodes. All war-fighters, drones, tankers and command centers are assumed to have computing, storage and transmission (for communication) capabilities which might be self-sufficient or insufficient to deploy blockchain technology in IoBT. In the case of insufficient resources, IoBT devices form a cluster based on their need to participate in Blockchain-enabled IoBT. To develop the simulator called FastChain [17], we used NS-3 and developed sharding enabled blockchain for IoBT. In other words, the networking part is implemented in NS-3 and sharding enabled Blockchain is developed on top of networking model of NS-3 where IoBT nodes are created for portraying the IoBT devices/nodes. IoBT nodes have unique addresses and can establish communication link among them. Furthermore, IoBT devices which have insufficient resources to participate sharding enabled blockchain, form clusters based on their computing, storage and communication range (or mobility pattern) capabilities.

Fig. 10 shows how the IoBT nodes in the battlefield scenario are connected through either single-hop or multi-hop links. Those IoBT nodes are mapped with networking nodes in NS-3 in the network layer, as shown in Fig. 10. Finally, those connected nodes build the blockchain

ecosystem by participating in blockchain either by individually if they have sufficient resources (such as computing, storage and communication) or by forming groups if they do not have sufficient resources (computing or storage or communication) to participate in blockchain-enabled IoBT. As shown in Fig. 10, upper level illustrates the blockchain implementation where IoBT nodes are assigned some responsibilities.

3.2.3 Building Blocks of FastChain Simulator

This section presents the architecture and building blocks of FastChain simulator that was written in NS-3, as shown in Fig. 11. The 'Main' program uses seven classes represented in the oval rectangle, as shown in Fig. 11, and creates the IoBT network of battlefield nodes and identifies possible shard for the required task in IoBT scenario. The main implementation calls all the classes and generates IoBT devices and places in a given location randomly. Each IoBT node gets random values for computing, storage, and transmission range. The networking connectivity for the nodes is created in the main program. Specifically, three types of IoBT nodes representing tankers, drones, and war-fighters are randomly created with resources (computing, storage and communications) and placed in the battlefield scenario. On top of these IoBT nodes, there will be a command center as shown in Fig. 11. Connectivity among IoBT nodes is created using networking modes available in NS-3. All IoBT nodes are visualized in NetAnim map for their visual representation. Different colors represent different IoBT devices. In particular, a blue-colored node represents tankers, red color represents drones, and the green represents the ground war-fighters (as shown in Fig. 12). Furthermore, the size of the IoBT node represents the proportionality in terms of its computing, storage, and transmission capacities.

The unique height and width for each IoBT node is computed using the function detailed in *Algorithm* 1. Each node has unique height and width which are proportionality to their computing, storage, and transmission capabilities, as explained in *Algorithm* 2. Once all nodes are initialized with their respective capacities in terms of storage, computing and transmission range, the size of the node, as shown in Fig. 12, is determined for visualization purpose. We briefly describe the different IoBT nodes and their equivalent class of FastChain simulator in NS-3 in Fig. 11.

Algorithm 1 Function returning two dimensional array [height][width] for a node

1: **procedure** FINDNODESPEC(v, u, p)
2: $r \leftarrow twodimensionalarray$
3: **for** $i \leftarrow 1, n$ **do**
4: height = base + offset * $\frac{v(i)+u(i)+p(i)}{max(v)+max(u)+max(p)}$
5: width = base + offset * $\frac{v(i)+u(i)+p(i)}{max(v)+max(u)+max(p)}$
6: $r[0][i] \leftarrow height$
7: $r[1][i] \leftarrow width$
8: **end for**
9: return r;
10: **end procedure**

Algorithm 2 Algorithm for creating dynamic node sizes

1: Create *num* number of random numbers between the threshold defined where *num* is randomly generated number or entered through command line by a user
2: Create a vector and push the numbers into a vector
3: Repeat steps 1 2 until three vectors for storage, computing, and transmission are filled respectively
4: Compute the height and the width for each node depending on the computing, storage and transmission range factor of the node by calling the function FINDNODESPEC(v, u, p)
5: Create the number of respective nodes as entered through command line or randomly generated
6: Setup Point to Point or Wi-fi connections between the nodes
7: Simulate the nodes using NetAnim with the height and width calculated from the function

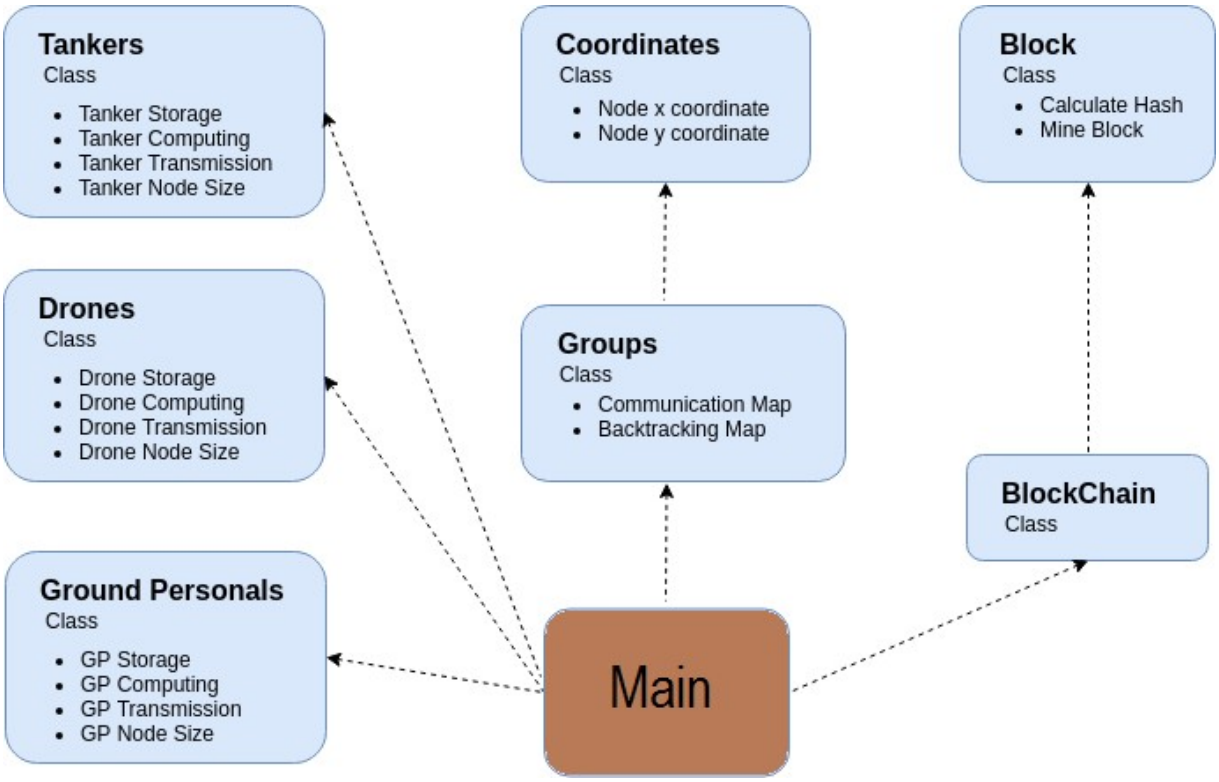


Figure 11: Class diagram of FastChain simulator [17].

A. *Tankers*

The *Tankers* class is responsible for handling all the requirements for tankers in IoBT. Specifically, *Tankers* possess the following properties.

- Tanker's Minimum Storage Capacity

- Tanker's Maximum Storage Capacity
- Tanker's Minimum Computing Capacity
- Tanker's Maximum Computing Capacity
- Tanker's Minimum Transmission Range
- Tanker's Maximum Transmission Range
- A two-dimensional array containing height and width for every Tanker node

The *Tankers* is a class which expects user input for the number of tankers they want in the battlefield context for IoBT. Hence, the main adds the same amount of nodes requested by the user in IoBT scenario. Similarly, *Tankers* class asks users for the minimum and maximum capacity of storage, computing, and transmission values for each Tanker node. A random number is generated between the minimum and maximum value as requested by the user for each of the node's respective capacities for storage, computing and transmission range. All the values generated are pushed back into respective vectors for storage, computing, and transmission range. The first values in these vectors represent the root node while other values represent the added Tanker nodes.

Alternatively, users have an option to automate the program, in which case, a random number of IoBt nodes for Tankersares between 1 and 3 automatically without asking users to input the range. Random numbers generated for the each node's capacities range from 10 - 1000 in the case of automation.

B. *Drones*

The *Drones* class has precisely the same functionality as the class *Tankers*. The *Drones* class is responsible for generating all the necessary variables needed for the Drone nodes. Similar to *Tankers*, *Drones* take user input to generate the storage, computing, and transmission for every Drone nodes and the desired number of Drones in the simulation. For automated simulation process, Drones automatically generate these numbers in the beginning. In this case, drones are generated from from the range [1, 3] and the capacities of Drone can fall from [10, 1000] units of respective capacities. The node sizes are generated and filled in the two-dimensional array as it is done for the *Tankers*.

C. *Ground Personals or War-Fighters*

The *Ground Personals* class also inherits the same functionality as *Tankers* or *Drones* class. *Ground Personals* get the required user input for the number of Ground Personals and generate random numbers in between the minimum and maximum numbers representing storage, computing, and transmission capacities of Ground Personal nodes. It computes dynamic sizes of the IoBT nodes using the same algorithms, *Algorithm 1* and *Algorithm 2*.

D. *Coordinates*

Coordinates class generates geolocations of the nodes for their positions in the NetAnim. The functions in *Coordinates* generate random numbers for the x and y values on the NetAnim graph. These numbers are then stored in a variable using another function for later use. These coordinate values are used to find the Euclidean distance between the IoBT nodes, along with the transmission range, to find out whether given IoBT nodes can directly communicate or not. The vector of tuples hold the values for the coordinates.

E. *Groups*

Groups is the backbone class in the FastChain as this class selects the group of minimum nodes with the capacity required to fulfill the storage, computing and communication requirements for sharding enabled Blockchain.

Algorithm 3 explains this process in detail on how to form a group based on their needs and capacities to precipitate in blockchain-enabled IoBT. Using the Transmission vector generated initially, we then design the communication model to map the nodes to the ones it can communicate to. Two IoBT nodes can communicate directly with each other if the euclidean distance between the two nodes is less than the transmission range of both of the nodes or there is a line of sight. Hashmap is created to store the mapping between a given IoBT node and the IoBT nodes it can communicate with. Every node is a key in the Hashmap with its values being possible shard nodes. The x and y coordinates of each node from the cotuple vector are used to find the Euclidean distance. We start with the first node and calculate the Euclidean distance with the next node and check if it is less than the transmission range of both the nodes. If so, we add it to the hashmap. We access every node and compare it with every other node and create the Hashmap or the communication model. In the algorithm, for a given size say k, we check if the vector current is the same size as the k. If yes, we check if the combination can fulfill the storage and computing requirement for the given task. If the combination does fulfill the task requirement, we return the found combination as the best combination. If it does not fulfill the requirement, we keep on checking the other combinations of the same size iteratively.

Furthermore, if the vector current is not yet the size of k, we add on further nodes. We select the first node in the given current vector, which is 0 initially. We select all the values with the first node as the key from the hashmap. We then select the values from the given index to the end of the values. Inside the loop of the selected values from the hashmap, we check if the element selected is in the keys of all the elements of the current vector. If yes, add the IoBT node to the current vector. If not, we continue with the loop.

If the node is added to the current vector, we recursive call the function backtrack. If the storage and computing requirement for the task is fulfilled, we return the vector. If the requirement not fulfilled, we pop up the last element from the current vector and check for the other combination of the same size.

Algorithm 4 depicts how we find the best shard for blockchain-enabled IoBT. The algorithm first checks if any individual nodes can complete the given storage and computing requirement of the given task. If not, we start with the first loop which is the loop of integers starting with 2 to the total number of nodes. The first loop, in *Algorithm* 4, represents the size of the node combinations we are looking for. The second loop selects every node as the first node and checks for all the combination of size k and calls the function backtrack. This way, we always look for the smaller size of combination of nodes first. We check if all the smallest combination possible that can communicate with each other can fulfill the job requirement. If the algorithm does find such a combination, it returns the vector and breaks from the function. If not, it keeps on searching for best-suited combination.

F. Block

The *Block* class in the program has the index value, nonce value, Hash value, the required data to create a block, and the current time as the primary property. *Block* class does the entire block implementation on the best combination of nodes we found from the algorithms above. *CalculateHash* function generates the Hash value from the sha256 hash value generating script. These values are sent as a input stream to the hash generating script to generate the hash vales: Index << current time

<< nodedata << Nonce value << Hash value of the previous block. MineBlock function in the class checks if the generated hash value is the correct one according to the difficulty level set before mining the block where difficulty level refers to the complexity of consensus protocol. The first *ndifficulty* strings in the hash value should be 0 to meet the difficulty to mine the block. The *Block* class generates the right hash value for the block given the input as the combination of nodes we found as the best one to fulfill the task.

Algorithm 3 Algorithm for grouping IoBT nodes for clusters

```
Vector couple tuples containing coordinates of n number of nodes
Vector Transmission range integer of total n number of nodes
Create hashmap() to list possible grouping nodes for each node
Generate n number of random integers x and y
Push every (x,y) as a tuple to the vector cotuple
for i 1, n do
    for j 1, n do
        if euclideandistance <= min(radius node1, radius node2) then
            Add the node to the vector
        end if
    end for
Update the hashmap with possible grouping nodes by evaluating the computing and storage capabilities of each node
```

Algorithm 4 Algorithm to find the best shard

```
if If any of the individual node can complete the given storage and computing requirement of the given task then
    Return that node and break
else
    for k 2, n do
        for i 0, n do
            Set vector current to 0
            Call the function backtrack
            As soon as the combination with the minimum node fulfills the requirement for the task, it breaks from the loop.
        end for
    end for
end if
```

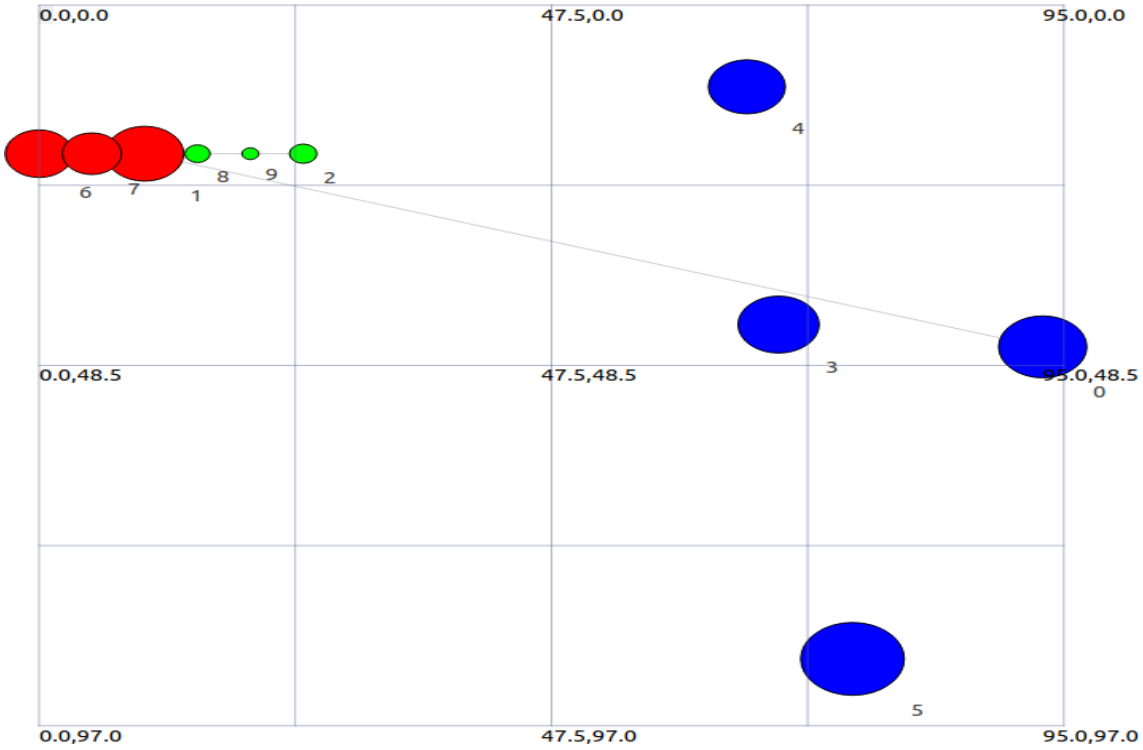


Figure 12: Visualization (Example Scenario) of IoBT nodes in FastChain Simulator

G. *Blockchain*

The *Blockchain* class simply creates the vector to place the block that has been mined. If there are no previous blocks in the blockchain, it generates the initial genesis block for the blockchain. The mined block is then pushed in the vector of the blockchain.

3.2.4 Analysis and Discussion

As noted earlier, in FastChain simulator, we have modular C++ script written in NS-3 where the main program calls different functions and classes to create IoBT scenario to simulate the same. IoBT nodes, when they don't have enough resources in terms of computing and storage, are grouped into clusters according to their storage capacity, computing capacity, and transmission range. In FastChain, IoBT nodes communicates with each other using peer-to-peer communications to exchange information among Tankers, Drones, and Ground Personals. IoBT nodes are generated randomly either based on user input or automatically based on default values set in FastChain simulator. Then, each node get randomly generated values for storage, computing and commutation range values from their respective ranges specified by the user or default set in the program. Algorithms 1 through 4 are used to increase the size of the IoBT nodes proportional to their resources and create a communication model based on the geolocations and transmission ranges of IoBT nodes. Based on the communication ranges of IoBT nodes, the communication ranges are used to

find the combination of a suitable set of IoBT nodes to fulfill needs where all nodes in the cluster can communicate with each other to participate in IoBT. If the IoBT device such as Tanker has enough resources to participate in blockchain-enabled IoBT, it can participate alone without being a member of the cluster. Specifically, the hashmap is used to find which IoBT node can talk to which other IoBT nodes. In FastChain simulator, every node has the list of nodes to which it can establish a connection in terms of transmitting range. Once each node has the hashmap, each node updates the hashmap to make sure these nodes can fulfill not only communication but also computing and storage capabilities required for sharding enabled IoBT. Once all nodes, either can participate individually or through formed clusters, use the input streams to create the block and add to the blockchain.

The IoBT scenario in FastChain can be visualized through the NetAnim Animator which shows IoBT nodes and their locations and sizes. Visualization can also help to see whether these IoBT nodes can communicate with each other or not. Fig. 12 shows the colored nodes where blue-colored nodes represent the tankers, red-colored nodes represent the drones, and the green-colored nodes represent ground personals/war-fighters. The simulation can be run where we can see how IoBT device can interact with other IoBT devices, and send and receive the information. This allows us to select suitable nodes that can interact and work efficiently together for sharding enabled blockchain system in the battlefield context. FastChain shows how the limited resources during the battlefield scenario can be efficiently used through collaboration by grouping them dynamically.

3.2.5 Conclusion

In this work, we developed and presented a simulator, called FastChain, built in NS-3 which simulates the battlefield scenarios with military applications which connects tankers, soldiers, and drones to form IoBT. The simulator uses the sharding enabled blockchain for trustworthy IoBT operations. Resource constraint IoBT devices form a group to participate in sharding enabled blockchain for IoBT scenarios. Researchers, educators and policymakers working on IoBT or similar scenarios can use the FastChain simulator and evaluate their systems.

3.3 Scalable Blockchain Solutions

Blockchain-based audit trails provide a consensus-driven and tamper-proof trail of system events that are helpful in creating provenance in enterprise solutions. However, taking into account the transaction bulk generated by these applications and the throughput limitations of existing blockchains, a single ledger for record keeping can be inefficient and costly. To that end, we see an imperative need for a new blockchain design that is capable of addressing current challenges, without compromising security and provenance. Hence, we propose *BlockTrail*, a scalable and efficient blockchain solution for auditing applications. *BlockTrail* fragments the legacy blockchain systems into layers of co-dependent hierarchies, thereby reducing the time and space complexity, and increasing the throughput. *BlockTrail* is prototyped on “Practical Byzantine Fault Tolerance” (PBFT) protocol with a custombuilt blockchain. Experiments with *BlockTrail* show that compared to the conventional schemes, *BlockTrail* is more efficient, and has less storage footprint.

3.3.1 Scalable Blockchain Platform for Auditing

Audit trails are important for efficient record management and provenance assurance [19], [20]. For example, government agencies are responsible for appraising properties and collecting taxes from residents [21], [22]. Starting from cities, these agencies work at various levels, including counties, states, and federation. As such, they keep track of property exchange, tax collection, permits, etc. Furthermore, these agencies have applications that generate audit trails to perform auditing and ensure system transparency. These applications continuously monitor the application's database, and generate an audit trail record upon change in the value of an object. However, due to the client-server relationship, audit trails are vulnerable to a single point-of-failure, whereby an adversary can externally and internally manipulate database and audit trails.

An intuitive solution to safeguard audit trails from single point-of-failure is to replicate them over all applications. This will raise the attack cost for the adversary since corrupting audit trails would require attacking all applications. This replication of audit trails can be achieved using blockchains, to enable secure, transparent, and immutable management of audit trails without needing a trusted intermediary [23], [24].

Current blockchain systems operate with a single-ledger shared among all system entities. The use of a single-ledger is therefore considered as a baseline model, atop which all applications abstract their services. However, the use of single ledger not only increases the storage footprint but also creates a bottleneck by preventing parallel processing. Applied to auditing, blockchains systems suffer from enormous space and time complexity, owing to the rate and size of transactions.

To address those challenges, we take a clean-slate approach towards the architecture of blockchain systems. We propose a multichain blockchain model that segregates the network into a set of layers, each capable of processing transactions independently. We leverage the hierarchical structure of eGovernment applications to facilitate parallel transaction processing and subsidized storage overhead. Moreover, the layered architecture also increases the throughput of the system by reducing congestion and transaction stall. In addition to the layered architecture, we further enhance capabilities of *BlockTrail* by using "Practical Byzantine Fault Tolerance" (PBFT) as the consensus protocol. In contrast to the existing schemes such as Proof-of-Work (PoW) and Proof-of-Stake (PoS), PBFT is energy efficient and achieves higher throughput.

A major limitation of PBFT is its lower fault tolerance compared to PoW and PoS. While PoW and PoS can withstand up to 49% malicious entities in the system, PBFT, on the other hand, can only sustain $\approx 30\%$ malicious nodes [25]. This is one of the reasons why PBFT has not been popular among applications with weaker trust models. However, specific to the requirements of our audit trail application, we take sufficient measures to equip *BlockTrail* with strong security measures in order to mitigate various attacks.

Contributions. We make the following contributions in this paper 1) We revisit the legacy designs of blockchains, and introduce a multilayer blockchain architecture that ensures higher

throughput with low processing delays. 2) We present *BlockTrail*; an end-to-end blockchain solution for audit trail applications that uses the multichain blockchain model to provide secure and tamper-proof audit trails. 3) We provide the theoretical constructs of *BlockTrail* and validate its performance through experiments and simulations.

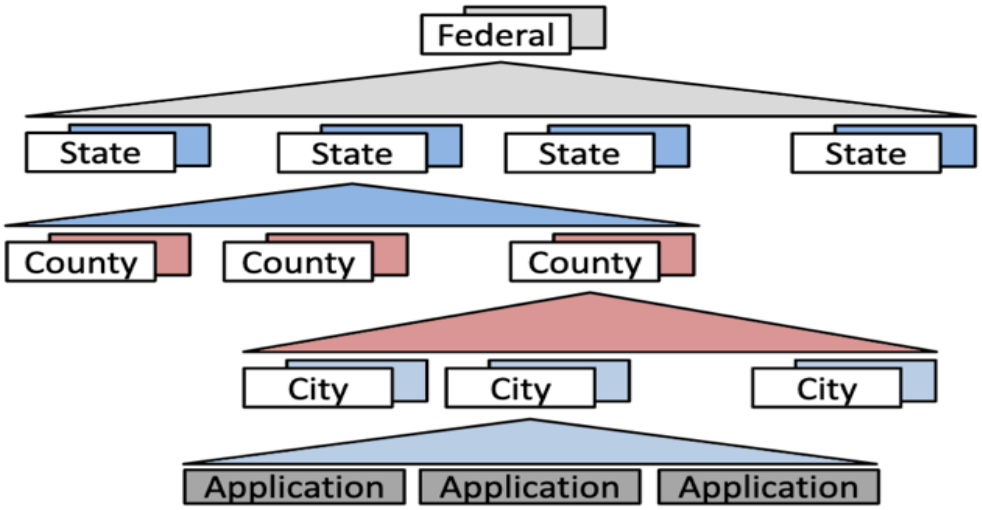


Figure 13: Multichain blockchain system design tailored to the specifications of *BlockTrail*.

3.3.2 BlockTrail Design

We begin by providing an overview of a audit log application that we use for the instrumentation of *BlockTrail*. The first step in design is the access to a large-scale audit log generation system that is currently being used by an enterprise. For this purpose, we used the services provided by ClearVillage Inc. [26], which provides software for cities and counties.

A. System Architecture

As defined previously, audit trails generated by the application are broadly associated to the exchange of property information among multiple entities at different hierarchies. This exchange of information occurs among: 1) peers (replicas) within the same city, 2) cities within counties, 3) counties within states, or 4) states within a country.

In conventional schemes of generating blockchain-based audit trails, a global blockchain is used to incorporate all the transactions. Although, this serves the purpose of secure and tamper-proof audit trails, it is not efficient and scalable. Each transaction has to traverse the entire network and get approval from all the peers. In particular, a local transaction related to a state change at the city level will require approval from all other parties in other cities that might not be relevant to that transaction. In addition to causing delays, this also limits the system throughput, since PBFT protocol serializes the transaction processing.

We argue that efficiency and throughput constraints faced by conventional systems can be

resolved by partitioning the network into multiple hierarchies. As such, transactions that are specific to a group of organizations within a city must be processed locally, while the transactions related to cities within a county can be processed at the county level and stored in county's blockchain. Taking this bottom-up approach from peers within the cities to the states within a country, we obtain a hierarchical tree of blockchain system that incorporates multiple blockchains, each holding data of its corresponding set of peers. Transactions will be generated by the organizations within the cities that act as a root in the system. Each transaction will have an identifier that will determine its destination blockchain.

Using this structure, our system will be able to achieve the following features: 1) Transactions within the same branch can be processed in parallel, thereby enabling parallel processing and increasing throughput. 2) For a transaction within same branch, the approval will be required from the leaf nodes within that branch that are relevant to the transaction. This will reduce the processing overhead incurred by transactions in conventional scheme. 3) Other than transaction generation and processing, this scheme is highly efficient in blockchain queries during auditing process or for conflict resolution. In Figure 1, we show the topology of our hierarchical blockchain paradigm, and in the following, we provide the notations that capture the abstraction of our model. Let $L_f = \{L_1, L_2, \ldots, L_s\}$ denote the country-level (federal) hierarchy that incorporates a set of all states within the country. This hierarchical blockchain paradigm can be extended from four levels to k levels, to increase the scallibility and reduce the time and space complexity. Keeping in mind the baseline fault tolerance of PBFT, we assert that the minimum number of replicas in blockchain, at each level is $s \geq 4$. For each state in L_f, let $L_i = \{l_1, l_2, \ldots, l_c\}$ be a set of counties in state. For each county in L_i, let $l_j = \{p_1, p_2, \ldots, p_d\}$ denote the number of cities that are associated with each county. Finally, for each city in l_j, let $a_q = \{n_1, n_2, \ldots, n_r\}$, be the set of peers (audit log applications), operating within the city. Given this topology, the overall size of the network S, determined by the number of audit-log applications, can be computed using.

$$S = \sum_{i=1}^{s} \sum_{j=1}^{c} \sum_{q=1}^{d} X_{qji} \tag{1}$$

Here, X_{qji} represents the position of a node within the system (identified by city, county, and state indexes). For each level, we have primary replica that executes the verification process. Specific to the design outlined in this paper, we have a primary replica for each city, county, and state in the system. Therefore, the total number of primary replicas in our blockchain system are $d + c + s$.

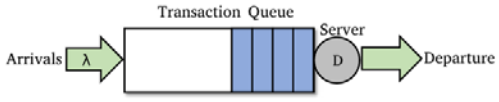


Figure 14: *M/D/1* queue

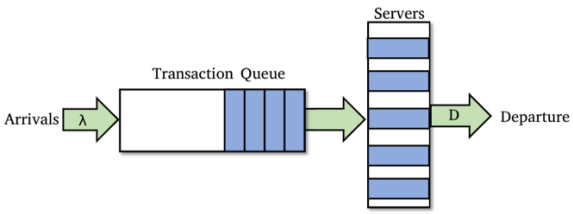


Figure 15: *M/D/c* queue

3.3.3 Blocktrail Analysis

A. Transaction Processing and Throughput

To understand the efficiency our system with respect to the transaction processing, we use a Markovian model that broadly formulates the PBFT-based blockchain systems. To that end, we envision that the system can be viewed as a Poisson process characterized as an *M/D/1* queue at the primary replica [27]. Here, *M* denotes the arrivals determined by a Poisson process, *D* denotes the deterministic mean service time, and 1 shows that there is one server in the system. In *M/D/1* queue, as shown in Figure 14 where transactions are arriving with rate λ, and there is one server that process the transactions at the average rate *D,* λ denotes the mean arrival rate of the transactions at the primary replica and *D* denotes the mean service rate of the active replicas that collectively act as a server. From this, we can derive $\rho = \lambda/D$, which denotes the utilization of the server. If the arrival rate is less than the service rate $\lambda \leq D$, there is no queuing at the primary replica, and each transaction gets processed before the next arrival.

However, in practice, the rate of incoming transactions is usually greater than the rate of transaction confirmation [28]. Therefore, this leads to the formation of a queue at the primary replica. In PBFT, if there are *a* number of active replicas in the system, then the minimum number of messages exchanged to verify the transaction are $a(a-1)$. Assuming that the time taken to exchange one message in the system is *t*, then the total time *T* taken to process a transaction becomes $t(t-1)$. Therefore, as the size of network grows and the number of active replicas increase, the time taken to process the transaction decreases, and the service time of server *D* decreases. Some key performance indicators of *M/D/1* queue are the mean number of transactions in the system and the average wait time for each transaction. As such, the mean number of transactions *L* in the system can be calculated as:

$$L = \rho + \frac{1}{2}\frac{\rho^2}{1-\rho} = \lambda t^2 - \lambda t + \frac{1}{2}\frac{(\lambda t^2 - \lambda t)^2}{1 - \lambda t^2 - \lambda t} \tag{2}$$

In (2), ρ is the server's utilization, λ is the arrival rate, and *t* is the time taken to exchange one message. Moreover, the average wait time for a transaction in the system *w* is:

$$w = t^2 - t + \frac{1}{2}\frac{\lambda(t^2 - t)^2}{1 - \lambda t^2\ \lambda t} \tag{3}$$

Applied to our multichains, the total number of servers increase at each layer for parallel processing. When the number of servers increases, the number of replicas splits between the servers, thereby reducing the service time for each transaction. In such conditions, the system reflects an *M/D/c* queue presented in Figure 15 where with transactions arriving at mean rate λ, and a group of servers are processing transactions with rate *D*. Here, *c* depends on the total number of replicas related to the transaction. For instance, lets assume a transaction tx_1 that is initiated between two cities C_a and C_b at time $t1$. The total replicas involved in the verification process are $a + b$. On the other hand, another transaction tx_2 is initiated at the same time t_1 among two different cities C_e and C_f, having total active replicas $e + f$. Now, these two transactions can be processed in parallel if the following condition is met:

Condition 1: Two transactions can be considered to be non-overlapping if their associated

active replicas are unique and have no intersection. $(C_a \cup C_b) \cap (C_e \cup C_f) = \emptyset$.

Depending on the size of replicas, each transaction will be processed accordingly. Under the assumption that at a given moment, there is a set of size c replicas that satisfy the aforementioned criteria, the system will behave as an *M/D/c* queue for transactions destined for each server. As the size of server may vary, depending on the number of verifiers involved with the transaction, the verification time and the throughput of the system may also vary accordingly.

B. Complexity Analysis

A key challenge with blockchain-based audit trails is the time and space complexity associated with the network and the blockchain size. The time complexity involves the time taken by peers to develop consensus over the blockchain state. The space complexity involves the storage and the search overhead that compounds due to append-only blockchain design. We suggest that the multilayer architecture of *BlockTrail* can be helpful in reducing the time and space complexity to achieve faster consensus and enhance storage capability of peers. For this analysis we used 100 peers, federal level transaction are 0.1%, state level are 0.9%, county level are 9% and city level are 90%. Consensus time is measured in microseconds presented in Figure 16.

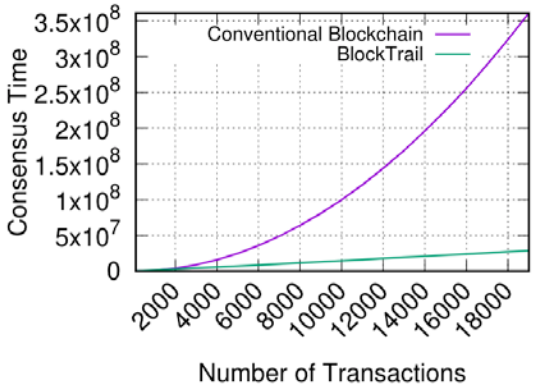


Figure 16: Complexity Analysis of *BlockTrail*.

To estimate the complexity of the system, let the total number of transactions in the system be $_bt$. Let t_d be the total number of transactions at the city level, t_c be the total number of transactions at county level, t_s be the total number of transactions at the state level, and t_f be the total number of transactions at the federal level. In (4), we show the relationship among these transactions. We assume that most transactions are exchanged at the city level, and the amount of transactions decreases as the hierarchy increases.

$$\hat{t} = t_d + t_c + t_s + t_f \quad where\ t_d \gg t_c \gg t_s \gg t_f \tag{4}$$

$$t_d = \sum_{i=1}^{d} t_d i, \quad t_c = \sum_{i=1}^{c} t_c i = \frac{1}{2^{\alpha}} t_d \quad t_s = \sum_{i=1}^{s} t_s i = \frac{1}{2^{\beta}} t_d$$

$$t_f = \sum_{i=1}^{f} t_f i = \frac{1}{2^{\gamma}} t_d, \quad \hat{t} = \left(1 + 2^{-\alpha} + 2^{-\beta} + 2^{-\gamma}\right) t_d \tag{5}$$

1) *Space Complexity: BlockTrail* reduces the space complexity of system by optimizing the transaction overhead at each layer. Storage used by a conventional blockchain system is ${}_bt \times n$ where ${}_bt$ is the total number of transaction and n is the number of peers. Since ${}_bt >> n$ the space complexity of flat blockchain system is $O({}_bt)$. However, a major downside of this method is that every peer is required to maintain a log of transactions that may not be related to its application. Benefiting from the hierarchical structure of *BlockTrail* and the non-overlapping nature of transactions, we suggest that the space overhead can be considerable reduced.

In our design, a major fraction of transactions is stored at the city level. Since transactions particular within the city are only stored locally, all other cities are not required to participate or store transactions. Deriving from (5), the major fraction of city transactions can be be computed as follows:

$$t_d i + \left(2^{-\alpha} + 2^{-\beta} + 2^{-\gamma}\right) t_d \tag{6}$$

Since $t_d i$ is the dominant component, therefore, the amortized cost of storage, outside the city layer becomes negligible, and the complexity of the system approximates to the complexity at the lowest layer.

$$\begin{gathered} \left(2^{-\alpha} + 2^{-\beta} + 2^{-\gamma}\right) t_d \approx \varepsilon \\ O\left(\left(1 + 2^{-\alpha} + 2^{-\beta} + 2^{-\gamma}\right) t_d\right) \approx O\left(t_d i\right) \end{gathered} \tag{7}$$

2) *Time Complexity:* With respect to time, we explore two aspects of complexity namely, consensus complexity and search complexity. We show that by design, in *BlockTrail* amortized cost of search and consensus is better than the conventional blockchain.

Consensus Complexity. To achieve consensus over the state of blockchain with n replicas, $n^2 - n$ messages are exchanged. Assuming that the system receives τ number of transactions, the cost of consensus in the conventional blockchain model becomes $O(\tau^2)$. However, in *BlockTrail*, the system is partitioned into sublayers, each comprising of different number of replicas. This partitioning of the system, as shown in Figure 13 where levels denote the hierarchies that keep blockchains and at the lowest level, there are applications connected to a city that emanate transactions from audit trails, reflects a tree structure with branches depicting multiple layers. Leveraging the number of transactions at each layer and using (5), the cost of consensus in conventional (T_{conv}) blockchain and *BlockTrail* (T_{bt}) can be computed as:

$$\begin{gathered} T_{conv} = O\left(\hat{t}^2 - \hat{t}\right) \approx O\left(\hat{t}^2\right) \\ T_{bt} = \left(2^{-\alpha} + 2^{-\beta} + 2^{-\gamma}\right)\left(t_d^2 - t_d\right) \approx O\left(t_d^2\right) \end{gathered} \tag{8}$$

Since $T_{bt} \ll T_{conv}$, therefore, the cost of consensus in *BlockTrail* is much less than the conventional blockchains.

Search Complexity. Similar to the space complexity, the search complexity in blockchains depends upon the number of transactions that are logged at a particular layer in the system. As such, the search cost in conventional blockchains (S_{conv}), and *BlockTrail* (S_{bt}) becomes:

$$S_{conv} = O\left(\hat{t}\log\left(\hat{t}\right)\right) \quad S_{bt} = O\left(t_d \log\left(t_d\right)\right) \tag{9}$$

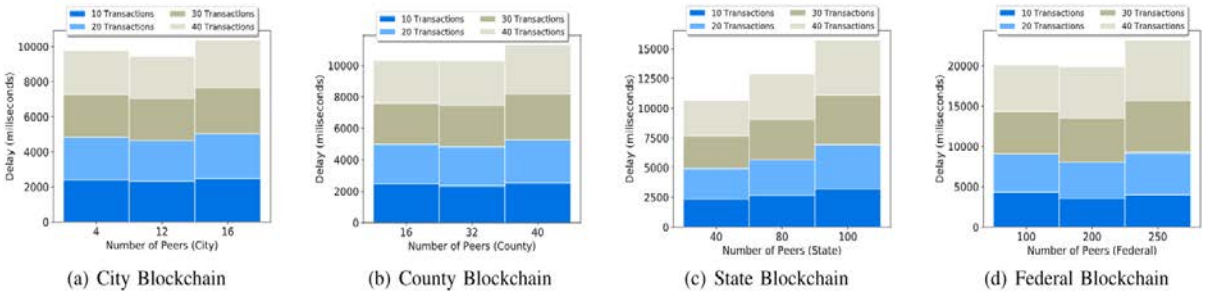


Figure 17: Results obtained from the simulations of *BlockTrail*.

Given $t_d << {}_b t$, therefore as shown in (9), the amortized search complexity in *BlockTrail* is much less than conventional blockchains. In Figure 16, we show the plots obtained by comparing complexity of conventional blockchains and *BlockTrail*. Our simulation results validate the theoretical analysis. Note that as the rate of transactions increases and the number of peers grow, the consensus time increases accordingly. The consensus time is smallest in the city blockchain and it increases as we proceed towards upper levels in hierarchy. Federal blockchain experiences the same delays as a conventional blockchain.

C. Security Analysis

We perform the security analysis of *BlockTrail*. We begin by outlining our trust model and adversarial model to access the strength of *BlockTrail* against various attacks. We use our analysis to suggest possible advancements that can be made to enhance the security of PBFT-based blockchain systems.

Trust Model. In *BlockTrail*, we assume that at any level of blockchain, there are four or more replicas that process a transaction. This criteria is critical for developing consensus in PBFT blockchains, that require approval from at least $3f+1$ active replicas in the presence of f faulty replicas. Since *BlockTrail* uses a permissioned blockchain, we can assume a more trustworthy environment where peers have mutual interests and limited incentives to misbehave.

Adversarial Model. For our adversarial model, we assume a computationally-bounded adversary that controls a set of malicious replicas in the system. We assume that the adversary attains the trust of other peers and positions himself among the active replicas. If the network has n replicas and the adversary controls f replicas, then in conventional blockchain design, the value of f has to be large enough ($n - f \leq 3f + 1$) to enable the adversary to attain control over the system. If the value of f is sufficiently large, then the adversary can compromise the system by asking the faulty nodes to withhold their signatures on a given transaction in order to halt the verification. In a layered design, a major challenge for adversary is to position his replicas in a way to obtain maximum benefits with minimum effort. In the following, we discuss the possible attack options for the adversary.

D. Positioning Malicious Replicas

The attacker with f malicious replicas can either randomly position them in the network at different layers of blockchain or select a targeted layer with fewer replicas to launch a targeted attack. In this section, we will evaluate both these design choices and analyze the state of the system under attack. First, we observe the possibility when the attacker randomly allocates f malicious peers in a layered blockchain system with b number of blockchains.

The random allocation of f replicas in b blockchains can be modeled as the classical *balls-into-bins* probability problem [29]. Provided that there are b blockchains and f malicious replicas, the probability that a replica gets allocated to any random blockchain is $\frac{1}{b}$. Using this premise, we are interested in answering the following questions: 1) Probability that two malicious replicas are allocated to a specific layer of blockchain, 2) Probability that a specific blockchain has exactly p malicious replicas, where $p \leq f$, 3) Probability a specific blockchain has no malicious replica. To answer the first question, let Alloc^k_i denote the event that i-th replica gets allocated to blockchain k, and let $\text{M}_{i,j}$ be the event that replica i and j get allocated to the same blockchain. By using Bayes' rule, we can find the probability of such an event as follows:

$$\begin{aligned}\mathbf{Pr}[\text{M}_{i,j}] &= \sum_{k=1}^{o} \mathbf{Pr}[\text{Alloc}_2^k|\text{Alloc}_1^k]\mathbf{Pr}[\text{Alloc}_1^k] \\ &= \sum_{k=1}^{b} \frac{1}{b}\mathbf{Pr}[\text{Alloc}_1^k] = \frac{1}{b}\end{aligned} \tag{10}$$

While doing the random allocation, a blockchain with more sensitive information may get exactly the number of peers that may compromise it. Let's assume that a specific blockchain b_s with p number of honest replicas cannot accommodate more than q malicious replicas. This leads to a problem raised in the second question which attempts to estimate the target blockchain gets exactly p malicious replicas, where $p \leq f$. This can be calculated by the following model:

$$\begin{aligned}\mathbf{Pr}[b_s \text{ has } p \text{ replicas}] &= \binom{b}{p}\left(\frac{1}{b}\right)^p\left(1-\frac{1}{b}\right)^{b-p} \\ &\leq \frac{b^p}{p!}\frac{1}{b^p} = \frac{1}{p!}\end{aligned} \tag{11}$$

It remains within the realm of possibilities that the attacker may not be able to position any malicious replica at any layer of the blockchain. This eventually adds to our trust assumptions of the system and may require less effort to defend against attacks. In the following, we show the probability that a specific blockchain in our system exhibits this property and contains no malicious replica belonging to the adversary. This addresses the third question above.

$$\mathbf{Pr}[\text{ blockchain gets no replicas}] = 1 - \left(\frac{1}{n}\right) \tag{12}$$

As the hierarchy of blockchain increases from city to county and eventually the federal blockchain, the security of the system also increases due to more active replicas being involved in the transaction confirmation. To enhance the security features of *BlockTrail* at lower levels of blockchain, we propose the following countermeasures.

E. Countering Targeted Attacks

In a situation where there are r number of honest replicas in a city blockchain and the attacker is able to position f faulty replicas such that $4f+1 > f+r$, then the attacker will be able to stop transactions verification in that layer. To counter this, we propose an expected verification time window W_t which will be set by the primary replica before passing the transaction to the verifying replicas. The primary replica knows the total number of active replicas in the system and can calculate the total number of messages to be exchanged until the transaction gets

verified. In this case, the total number of messages will be in the order of $(f + r)2 - (f + r)$. If one message, exchanged among $f + r$ peers, takes t0 time, then the total time taken for transaction verification will be $c \times (t02 - t0)$, where c is an arbitrary constant set by the primary replica. Based on these values, the primary replica can set an expected time window $Wt \geq c \times (t02 - t0)$ in which it expects all peers to validate the transaction and submit their response. Let tstart be the start time at which the primary replica initiates the transaction. If by Wt the primary replica does not receive the expected number of responses from the replicas, it will abort the verification process and notify the auditor.

Depending on the application's sensitivity, the primary replica can either set another optimistic value of Wt0, where $Wt0 \geq Wt$, and repeat the process or it can simply abort the process and notify the auditors in application regarding the malicious activity. We leave that decision to the audit log application and its sensitivity to malicious activities. However, in our experiments, we relax the condition of sensitivity and re-submit the transaction for another round of verification. We set a new expected verification time window Wt0 and wait for the response. Our choice of relaxing the condition of sensitivity is owing to the unexpected delays in the message propagation; given that our system would run over Internet. However, if the primary replica does not receive the approval of transaction the second time, it aborts the process and notifies the application.

F. Experiments and Evaluation

We first prototype BlockTrail on a popular blockchain framework called Hyperledger [30], to verify its correctness and consistency with blockchain systems. However, in Hyperledger, we do not have the flexibility of applying the layered blockchain design proposed in this paper. As such, we employ the core functionality of Hyperledger including orders (primary replica), replicas, and PBFT protocol. Leveraging the design constructs of Hyperledger, we proceed with abstracting its core functionality and developing our propriety blockchain system that is tailored to the specifications of our application. In the following, we outline the steps taken to deploy our propriety blockchain system.

For experiment, we used existing logs to generate the JSON packets to generate audit trail entries. These audit trail entries are generated by the application and sent to the relevant city, county, state or federal blockchain. The primary replica notifies all concerned replicas that are associated with a blockchain and request them to validate the transaction. We generate a series of transaction for each layer of blockchain. We vary the transaction rate by increasing λ, and note the time taken by all the peers to reach consensus over it. Additionally, for each layer, we vary the number of peers and the size of transaction to see the overhead in consensus time. λ was increased from 25 to 50, and the city peers were set to 10,20,30 and 50, the county peers were set to 50, 100, 150 and 200, and the state level peers were set to 80, 160, 240 and 320. Finally, the federal level peers were set from 100, 200, 300, and 400.

We evaluate the performance of BlockTrail by the time taken for all the nodes to reach to a consensus over the transaction sent by the primary replica. Let tg be the transaction generation time, and tc be the time at which it gets approval from all active peers and gets confirmed in the blockchain. In that case, the latency lt is calculated as the difference between tc and tg ($lt = tc - tc$, where $tc > tg$). The mean values of each experiment are plotted in Figure 17. It can be observed that as the number of peers increases at each layer, the consensus time increases considerably. Also, as the rate of incoming transactions increase, naturally, the consensus time increases. As expected, the time for consensus at the city level was less compared to county and the state level.

G. Related Work

We review work on secure audit logging mechanisms and contrast them with our approach to highlight our contributions. Audit Trails. Schneier and Kelsey [31], [32] proposed a secure audit logging scheme capable of tamper detection even after the system compromise. However, their system requires the audit log entries to be generated prior to the attack. Moreover, their system does not provide an effective way to stop the attacker from deleting or appending audit records, which, in our case is easily spotted by BlockTrail.

Waterset al. [33] proposed a searchable encrypted audit log, which uses identity-based encryption keys to encrypt audit trails, and allow search by certain keywords. Yavuz and Ning [34] developed a forward secure and aggregate audit logging system for distributed systems, without using a trusted third party. Zawoad et al. presented Secure-Logging-as-a-Service (SecLaaS) for storing virtual machine audit trails in secure manner, SecLaaS ensures confidentiality of users and protects integrity of logs by preserving proofs of past logs. Ma and Tsudik [35] looked into temper-evident logs that are based on forward-secure aggregating signature schemes.

Xu et al. [36] proposed to use game theory and blockchain to reduce latency by moving applications to edge servers. Similarly we are using the geographical proximity to store audit logs in servers that are close to reduce latency.

Blockchain and audit trails. Sutton and Samvi [37] proposed a blockchain-based approach that stores the integrity proof digest to the Bitcoin blockchain. Castaldo et al. [38] proposed a logging system to facilitate the exchange of electronic health data across multiple countries in Europe. They created a centralized logging system that provides traceability through an unforgeable log management using blockchain. Cucrull et al. [39] proposed a system that uses blockchains to enhances the security of the immutable logs. Log integrity proofs are published in the blockchain, and provide non-repudiation security properties resilient to log truncation and log regeneration. Chi and Yai [40] proposed an ISO/IEC 15408-2 Compliant Security Auditing system using Ethereum that creates encrypted audit logs for IOT devices. Chen et al. [41] proposed a Blockchains based system to address shortcomings in log-based misbehavior monitoring schemes used to monitor Certificate Authorities (CA). In contrast to prior work, BlockTrail is implemented by extending a data access layer of the business application, which only required modification to access layer, and no other modifications.

H. Conclusion

In this paper, we present BlockTrail; a multilayer blockchain system that leverages the hierarchical distribution of replicas in audit trail applications to reduce the system complexity and increase the throughput. BlockTrail fragments a single ledger into multiple chains that are maintained at various layers of the system. We prototype BlockTrail on an audit trail application and use PBFT protocol to augment consensus among replicas. We also propose new strategies to mitigate security risks associated with weak trust model of PBFT. Our experiments show that compared conventional blockchains, BlockTrail is more efficient with tolerable delays. In future, we aim to explore the application of BlockTrail beyond audit trails including IoT and health care.

3.4 Secure Blockchain Platform

Audit logs serve as a critical component in enterprise business systems and are used for auditing, storing, and tracking changes made to the data. However, audit logs are vulnerable to a series of attacks enabling adversaries to tamper data and corresponding audit logs without getting detected. Among them, two well-known attacks are "the physical access attack," which exploits root privileges, and "the remote vulnerability attack," which compromises known vulnerabilities in database systems. In this paper, we present BlockAudit: a scalable and tamper-proof system that leverages the design properties of audit logs and security guarantees of blockchain to enable secure and trustworthy audit logs. Towards that, we construct the design schema of BlockAudit and outline its functional and operational procedures. We implement our design on a custom-built Practical Byzantine Fault Tolerance (PBFT) blockchain system and evaluate the performance in terms of latency, network size, payload size, and transaction rate. Our results show that conventional audit logs can seamlessly transition into BlockAudit to achieve higher security and defend against the known attacks on audit logs.

A. Introduction

Enterprise business systems and corporate organizations maintain audit logs for transparent auditing and provenance assurance [42, 43]. In addition to their functional utility, the maintenance of audit logs is mandated by Federal laws. For instance, the Code of Federal Regulations of FDA, Health Insurance Portability and Accountability Act, etc. require organizations to maintain audit logs for data auditing, insurance and compliance [44]. Secure audit logs enable stakeholders to audit the systems' state, monitor users' activity, and ensure user accountability with respect to their role and performance. Due to such properties, audit logs are used by data-sensitive systems for logging activities on a terminal database. Often times, audit logs are also used to restore data to a prior state after encountering unwanted modifications. These modifications may result from attacks by malicious parties, software malfunctioning, or simply user negligence.

Audit logs typically use conventional databases as their medium for record keeping. Therefore, with databases, audit logs reflect a client-server model of communication and data exchange. The client-server model positions databases as a single point-of-trust for the audit logs, and therefore naturally a single point-of-failure. With this vantage of vulnerability, audit logs can be compromised in many ways. An adversary with root access to the database can manipulate critical information both in the database and the corresponding audit log. Once an audit log is compromised, the safety and transparency of the application is put to a risk. In the light of this weak security model, there is a need for secure, replicated, and tamper-proof audit logs that do not suffer from this shortcoming and have effective defense capabilities to resist attacks. To that end, we envision that blockchain technology can naturally bridge the gap to nicely serve the security requirements for audit log management, including ensuring security, provenance and transparency [45, 46].

Over recent years, blockchain has acquired significant attention due to its use in distributed systems [47]. In peer-to-peer settings, blockchain is capable of augmenting trust over an immutable state of system events [48]. The most prominent example of blockchain technology has been realized in Bitcoin [49]; a peer-to-peer digital currency that enables secure transfer of digital assets without the need of a trusted intermediary. Since Bitcoin, the use of blockchain has become prevalent in various applications and industries including smart contracts [50, 51], communication systems [52], health care [53-54], Internet of Things [55-56], censorship resistance [57], and electronic voting [58-59]. The potential of blockchains is fully utilized in an environment where, 1) entities belonging to the same organization have competing interests [60] and/or 2) there is a need for immutable data management whose security increases over time [61-62]. Because audit log applications meet the aforementioned requirements, they can intuitively use blockchain properties for an added security of audit logs. In Figure 18, we presented the audit log generation in an OLTP system where we annotate each step with a number to show the sequence of progression. Notice that the user generates a transaction to change the value from C to D, and the change is then recorded in the audit log by the database.

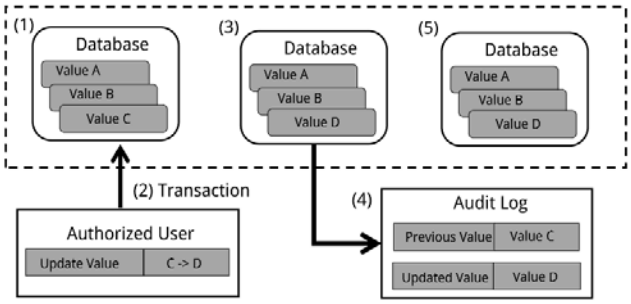


Figure 18: Audit log generation in an OLTP system.

Applied to an audit log application, blockchain can replicate the information contained in audit logs over a set of peers, thereby providing them a consistent and tamper-proof view of the system [63]. Blockchains use an append-only model secured by strong cryptography hash functions. The security of data in the ledger increases while the blockchain grows with time. Furthermore, a malicious party intending to compromise the system will have to change logs maintained by a majority of peers. This increases the cost and complexity of the attack and increasing the overall defense capability of the audit log application. However, the design space of blockchain is modular due to varying access control policies and consensus schemes. Therefore, it becomes a design challenge to apply suitable structural and functional primitives that best fit the application requirements and achieve the end goal of transparency and provenance. Motivated by that, we propose a blockchain-based audit log system called BlockAudit. Broadly speaking, in BlockAudit, we 1) capture system events generated by the data access layer of an enterprise application, 2) transform the acquired information into blockchain compatible transactions, 3) construct a peer-to-peer network consisting of entities that evaluate and approve the authenticity of transactions by executing a consensus protocol, and 4) lock the transaction in an append-only and immutable blockchain ledger, maintained by each network entity.

Contributions. In summary, in this paper we make the following key contributions: (1) We outline security vulnerabilities in audit log applications and discuss shortcomings of the prior work in addressing those vulnerabil-ities. (2) We present a blockchain-based audit log system called BlockAudit which addresses these vulnerabilities and ensures security, transparency, and provenance in the auditing system. Towards that, we review the modular constructs of blockchain systems and discuss suitable design choices that best fit the requirements of an auditing system. (3) We test the design of BlockAudit using a real-world eGovernment application, provided by Clearvillage Inc, and analyze its performance using three evaluation metrics, namely the latency, the network size, and the payload size. (4) Based on observations made from theoretical analysis and experiments, we discuss our proposed solution and provide future directions for research on blockchain-based audit logs.

B. Background and Threat Model

In this section, we provide the background of audit logs including their benefits and vulnerabilities. We also provide a threat model for the systematic exposition of the outlined vulnerabilities.

Audit Logs

An audit log is an essential component in online transaction processing (OLTP) systems such as order entry, retail sales, and financial transaction systems [64, 65]. The OLTP system maintains audit logs to monitor users' activity and provide insight into the sequential processing of transactions [66]. Each processed payment in OLTP system creates a unique record in the audit log. The aggregate volume of transactions and the total payment made during a financial year can be verified by consulting the data recorded in the audit logs. Moreover, these audit logs can also be used to identify discrepancies, anomalies, and malicious activities in payments. Audit logs have to be secure, searchable, and readily accessible from the application so that business users can easily view the chain of actions that lead to the current state of a business object. In Figure 18, we provide an overview of the OLTP system in which an audit log is generated once an authorized user commits a transaction to the database. The transaction makes a change in the value of an object and, as a result the change is recorded in the database and audit log. These changes can be matched later with the database and/or the application for auditing and provenance.
Benefits of Audit Logs. The audit log is widely used in modern information systems to provide a chronological record of changes being made to the data, and track the life-cycle of objects. Audit logs are also used to verify and authenticate operational actions, provide proof-of-compliance, ensure operational integrity, detect malicious activity, and provide system-wide provenance [67, 68]. Organizations that use audit log applications no longer maintain a paper trail for chronological record management, thereby saving cost and storage space with additional environmental benefits. With the elimination of the paper trail, the electronic audit logs are solely responsible for establishing security and the correctness of sensitive information. In the situation of an attack, the audit log is typically used as a starting point of forensic analysis.

Audit logs also contribute to organizing user behavior in applications. Since audit logs maintain the user activity over time and detect misbehavior, naturally, they promote responsible user behavior and reduce the chances of misconduct. The users remain aware of their actions being recorded in an audit log. Moreover, in the case of an attack or a malicious activity, audit logs can be used to ensure users are accountable for their actions.

The correctness of audit logs is imperative to find the cause of the attack and initiate suitable countermeasures. For example, the first step in identifying the solution to a system crash is obtaining appropriate knowledge of the conditions that lead to the crash. This knowledge can be obtained through audit logs which can be used to reconstruct the conditions of an event. Such reconstruction can correctly identify the root cause of the issue, such as network failures, system bugs, or information tampering. Furthermore, after fault detection, the system can be restored back to the original state by rolling back transactions to the point in time prior to the attack. Atop the real-time monitoring, audit logs can also be used to identify system-related problems such as implementation errors, software bugs, and deployment faults. Finally, audit logs can also help in intrusion detection, by providing useful information to detect unauthorized system access.

Vulnerabilities in Audit Logs. Despite the aforementioned benefits, audit logs are vulnerable to a series of attacks that may compromise the integrity of OLTP systems. An attacker can use multiple attack vectors which exploit the known weaknesses in OLTP systems and corrupt the state of the database and audit logs. Conventional schemes of protecting audit data include the use of an append-only device such as continuous feed printer or Write Once Read Multiple (WORM) optical devices. These systems work under a weak security assumption that the logging site cannot be compromised, which eventually keeps the integrity of the system intact. However, attackers have often exploited vulnerabilities at logging site to tamper with data in audit logs [69-70].

If the attacker acquires the credentials of an authorized user, he can corrupt the database as well as the audit log. On the other hand, if the attacker compromises the database by breaching its defense, he can manipulate the database and prevent it from populating audit logs. Then, not only he will be able to corrupt the database, but also disable the auditing procedure by blocking the backward compatibility of audit logs with the database.

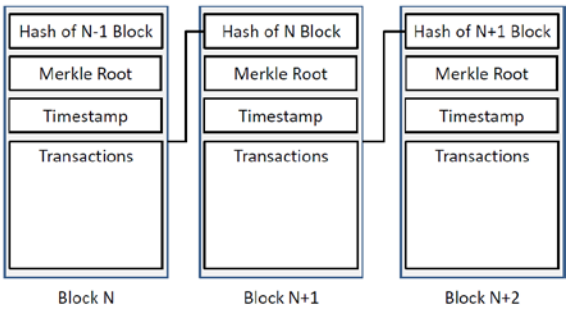


Figure 19: An overview of Blockchain structure consisting of three blocks.

Threat Model

To sufficiently analyze the vulnerabilities of audit logs and set the security model objectives, we present the threat model for the auditing systems in this section.
Inspired by the limitations found in the prior work [71-72], our threat model assumes an adversary that is capable of both physically accessing the trusted computing base (TCB) and remotely penetrating the OLTP system by exploiting software bugs. As such, the adversary can be a malicious third party aiming to tamper data to compromise auditing procedures. This would require the adversary to obtain root privileges to the system, or have significant knowledge of the system architecture. Additionally, the adversary can also hack and acquire the credentials of a root user of the system. This can be carried out using various attack procedures available in the conventional attack catalog [73]. However, possessing the knowledge of a private database system or a remotely acquiring credentials of a root user would require exceptional capabilities for the adversary. Therefore, we assume the third party attacker to have strong capabilities.
In a less hostile environment, the adversary can also be someone from within the system with root privileges. For instance, a corrupt auditor, who has tampered data for personal gains, might want to cover his act by changing data values. In contrast to the third party attacker, this adversary will not need sophisticated capabilities since he already has root privileges and the system knowledge.
For the system architecture, we assume an OLTP system similar to a retail sale repository. The system implements the design logic of an application using secure communication protocols such as SSL/TLS. Moreover, the system has a database that keeps records of sales and maintains a remote audit log. The audit log keeps track of the database changes through transactions, as shown in Figure 1. In such a design, the attacker can exploit the system by launching two possible attacks, namely the physical access attack and the remote vulnerability attack.
The Physical Access Attack. In the physical access attack, the adversary will use the root privileges to corrupt the database. As shown in Figure 1, the adversary will generate a series of transactions to change the values of objects in the database. Once the attacker manipulates the data, the database will automatically generate an audit log, tracking all changes made by the attacker. However, to evade detection, the attacker can either delete the newly generated audit log or modify its values. Furthermore, the attacker will also be able to tamper the history maintained by the audit log in order to corrupt the auditing process. Therefore, in the physical access attack, we assume an adversary inside or outside the system who has access to the key system components.

The Remote Vulnerability Attack. In the remote vulnerability attack, the attacker may only exploit the default vulnerabilities in the OLTP applications such as software malfunctions, malware attacks, buffer overflow attacks etc.. In this attack, the adversary, although not as strong as the physical access attack may still be able to contaminate the database and the audit log with wrong information. Despite these adversarial capabilities, we assume that the OLTP application is secure against the conventional database and network attacks such as SQL

injection and weak authentication. Generally, database systems used by corporate organizations are secure against these conventional attacks, and for the application service used in this paper, we ensure this requirement is meet.

C. Related Work

In the following, we review the notable work done in the direction of securing audit logging mechanisms. We also discuss the limitations of the prior work in light of the threat model (§2.2).

Audit Logs. Schneier and Kelsey [72-73] proposed a secure audit logging scheme capable of tamper detection even after compromise. However, their system requires the audit log entries to be generated prior to the attack. Moreover, their system does not provide an effective way to stop the attacker from deleting or appending audit records, which, in our case is easily spotted by BlockAudit. Snodgrass et al. [74] proposed a trusted notary based tampering detection mechanism for RDBMS audit logs. In their scheme, a check field is stored within each tuple, and when a tuple is modified, RDBMS obtains a timestamp and computes a hash of the new data along with the timestamp. The hash values are then sent as a digital document to the notarization service which replies with a unique notary ID. The ID is stored in the tuple, and if the attacker changes the data or the timestamp, the ID becomes inconsistent, which can be used for attack detection. Ray et al. [75] proposed a framework for maintaining secure audit logs in cloud computing platforms. In particular, their framework uses cryptography to maintain integrity and confidentiality while storing, processing, and accessing the audit logs. Ma and Tsudik [76] proposed a technique to generate an aggregate signature by sequentially combining individual log entry signatures using forward-secure, append-only signatures. This scheme provides provable security with efficient space utilization; where the correctness of individual entry can only be verified by generating the aggregated signature. Yavuz et al. [36] proposed a scheme that stores individual and aggregate signatures, where the storage of individual signatures increases the storage footprint while allowing individual verification of signatures.

Blockchains. A blockchain is a data structure that enables transparent and tamper-proof data management in distributed systems [78, 79]. As such, blockchain consists of a sequence of data blocks that are linked through on-way hash functions. Due to the one-way property of hash operations, blockchain exhibit the append-only model where once a data item is inserted it becomes immutable [80-81]. An illustration of the blockchain data structure is provided in Figure 19. Transaction ordering using blockchain is enabled by multi-party consensus schemes [82-83]. Popular among these schemes are the proof-of-work, proof-of-stake, and practical Byzantine Fault Tolerance [80,84]. Roughly speaking, a consensus algorithm is a set of instructions executed independently by each party in the system. The execution is completed if a majority under fixed bound obtains the same output from the computation. For more on blockchains and consensus schemes, we refer the reader to [85-86]. In Figure 19, we presented an overview of Blockchain structured consisting of three blocks. Notice that each block header consists of the hash of the previous block. This relationship gives blockchain, the property of an immutable ledger. Also notice that the merkle root ensures that the transactions are ordered in a sequence.

Blockchain and Audit Logs. Combining blockchain and audit logs, Sutton and Samvi [87] proposed a blockchain-based approach that stores the integrity proof digest to the Bitcoin blockchain. Bitcoin uses a proof-of-work (PoW) consensus protocol. As we show later in Table 3, PoW suffers from low throughput and high confirmation time. In particular, Bitcoin has a maximum throughput of 3–7 transactions per second. Therefore, for audit log applications that have a high transaction generation rate, the concept provided in [46] can be insufficient. Castaldo et al. [88] proposed a logging system to facilitate the exchange of electronic health data across multiple countries in Europe. They created a centralized logging system that provides traceability through unforgeable log management using blockchain. Cucrull et al. [89] proposed a system that uses blockchain to enhance the security of the immutable logs. Log integrity proofs are published in the blockchain providing non-repudiation security properties.

D. Problem Statement

The prior related research provides the groundwork for securing audit logs with blockchains and represent the foundation of our work. However, our major contribution is seen in our focus on audit logs related to enterprise business applications, focusing on scalability and performance. As outlined in §1, blockchain applications may vary in their access control policies and consensus schemes. Exploring the blockchain model for Enterprise business applications would require an understanding of their requirements, and methods to overcome the domain-specific design challenges, which we explore in this paper.

Another limitation that can be observed in [74, 89] is the inability to address Byzantine behavior among network peers. In other words, the application assumes all participating entities faithfully execute the consensus protocol without incurring any malicious behavior. However, in distributed systems adversaries can control a subset of replicas who can behave arbitrarily in order to withhold transaction processing and cause conflicting views among other replicas. Tolerance towards Byzantine nodes is a function of consensus schemes to be applied. For instance, permissionless blockchain applications such as Bitcoin can tolerate up to 50% of Byzantine nodes while maintaining operational consistency. On the other hand, PBFT-based private blockchains can tolerate only 30% Byzantine nodes. Therefore, the selection of a consensus algorithm can influence the security model of the application. In BlockAudit, we address the aforementioned limitations and present an end-to-end solution constructed by transforming knowledge problems into design problems.

Design Engineering

So far, we have discussed the benefits of audit logs, their key vulnerabilities, and the existing solutions that address those vulnerabilities. We have also presented a threat model to outline adversarial conditions. In this section, we use this knowledge to make design choices to meet the requirements of a practical blockchain-based audit log solution. In the following, we define functional, structural, and security requirements that we expect BlockAudit to meet.

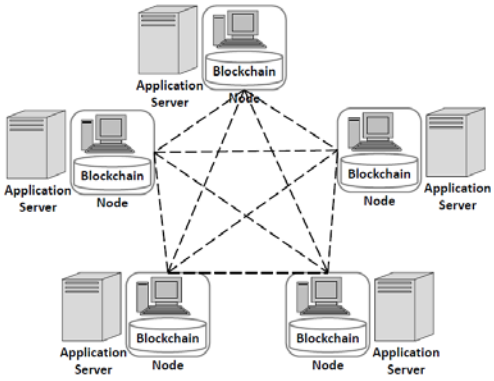


Figure 20: The network overview of nodes employing BlockAudit.

Functional Requirements. An audit log application is expected to ensure trust in the application data and provide tamper-proof evidence of transaction history when needed. Data tampering has to be prevented for the application data as well as the audit logs. However, a priority is given to the audit logs, since they are used to establish provenance. For this purpose, the audit log data should be stored across multiple peers in such a way that it remains consistent at each node, and therefore, hard to corrupt. If tampering happens at any node, the system should be able to detect and correct it. This requirement, however useful, comes with an assumption that a majority of peers behaves honestly, and faithfully executes the system protocols.

For audit data to be added to the blockchain, the participating peers in the audit log network must reach a quick consensus over a newly generated transaction. Since audit logs are generated in real-time and persisted inside the database transaction, therefore, any delay in using distributed audit logs adversely affects the system performance. In order to prevent such delays, the system needs to have low latency while maintaining the capability of processing a large volume of transactions. Additionally, the application should not add any data without consensus among a majority of peers.

The audit log system architecture should be modular and service-oriented so that it is possible for various types of applications to participate and benefit from this system. Moreover, audit logs should be data agnostic and must not rely upon the nature of data that is stored in them. The business application should be able to provide data in any format as per the requirements of the application.

Finally, the audit log system should provide searching and retrieval capability to enable the retrieval of any desired transaction or a set of transactions (e.g., audit log entries for the last ten minutes, all audit log entries registered against a specific user ID, etc.). The search needs to be fast and responsive to ensure the end user is able to perform the audit in real-time.

Structural Requirements. Keeping in view of the design the baseline models introduced [74,89], we envision that *BlockAudit* must operate in a distributed manner with application services running on multiple hosts without a central authority. As such each application peer would require its own blockchain node to become part of the the *BlockAudit* network. In Figure 20, the network overview of nodes employing BlockAudit. Notice that each node maintains an interface

that connects them to the audit log application. They exchange transactions with one another during the application life-cycle.

.The audit log system should have a high throughput and should be able to process a large number of transactions. *BlockAudit* should be able to support transactions of various sizes since the transaction size varies in audit log applications. The audit log system should be easy to integrate with existing system with minimal structural and functional changes in the application. It should also be independent of the underlying application database. Finally, the system auditing should be secure, transparent, and visible to all peers within the network.

Security Requirements. In the light of our threat model §2.2, we require *BlockAudit* to be secure in adversarial conditions. To that end, if the adversary launches a physical access attack, *BlockAudit* should be able to neutralize it and prevent data tampering at the source. If the adversary launches the remote vulnerability attack, *BlockAudit* should stop the attack propagation across the network peers. In other words, if the adversary

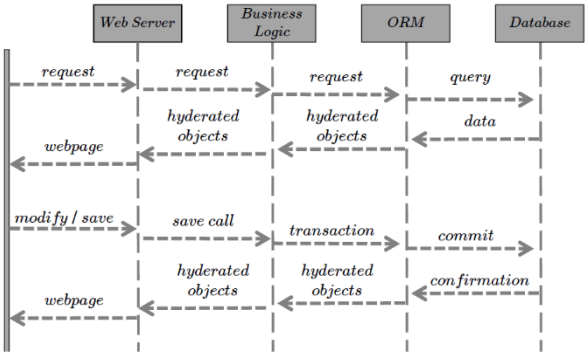


Figure 21: The information flow between various components of the application.

exploits a bug in the audit log of one peer, *BlockAudit* should immediately recognize the attack and notify the victim peer. Furthermore, the infection should be curtailed at the target zone, preventing its spread in the network.

In addition to the baseline attack model, we also expect *BlockAudit* to remain secure in the presence of Byzantine nodes. Therefore, if a strong adversary controls a subset of nodes in the network, he should not be able to corrupt audit logs or delay transaction verification. This can be achieved by either raising the attack cost i.e., constructing a large network or relaxing anonymity so that the adversary risks identity exposure by misbehaving.

E. **BlockAudit**

In this section, we show the implementation of BlockAudit. First, we describe the eGovernment application that we used to generate audit logs. Next, we show how a blockchain network is constructed to integrate audit logs. In that, we describe the methods of generating transactions, creating a distributed network, managing the access control, and developing consensus among peers over the state of the audit logs In Figure 21, the information flow between various components of the application is presented. Notice that the transaction is generated at the

business logic layer, and once the database commits to the transaction it is rendered on the web page.

Application Architecture

For *BlockAudit*, we use an eGovernment application provided by a company called ClearVillage inc which provides software solutions for various government operations e.g. property appraiser, building permits etc. The application uses a multi-tier system architecture comprising of web and mobile clients, a business logic layer, a data access layer, and a database. In the following, we describe the core functionality of each component along with its role in generating an audit log.

Web Applications. The web applications are built using asp.net and users access the application services through a web browser. Additionally, native clients are provided for Android and iOS, built using their respective development frameworks. The web application and web services are hosted on Microsoft's Internet Information Services (IIS) web server. The public side portal is available on the Internet and gives public users access to information without authentication. Atop this, a staff portal is provided to the organization staff, which is only accessible from within the organization, thereby providing another security layer.

Business Logic Layer. The business logic layer is an interface between clients and the database layer, responsible for implementing business rules. Among other functions, the business logic layer also manages data creation, data storage, and changes to the data with the help of object-relational mapping (ORM). Upon receiving a request from the client, the web server instantiates the relevant objects in the business logic layer, which uses the ORM to send the processed object to the client. The ORM writes changes to the objects in the RDBMS tables.

ORM. The ORM in the application provides a mapping mechanism that allows querying of data from RDBMS using an object-oriented paradigm [90-91]. Modern web applications are well suited for this technique since they are multi-threaded and are rapidly evolving. ORM also reduces the code complexity and allows developers to focus on business logic instead of database interactions. This application uses NHibernate[92]: an ORM solution for Microsoft .NET platform. NHibernate is a framework used for mapping an object-oriented domain model to RDBMS and it maps the .NET classes to database tables. It also maps Common Language Runtime (CLR) data types to SQL data types. The ORM inside a database layer creates a SQL statement to hydrate the object and passes it to the business logic layer. ORM also flushes the changes to the RDBMS and commits a transaction. Interactions between the application and RDBMS are carried out using the ORM. In Figure 21, we provide the information flow between application components.

Generating Audit Logs

In this section, we show how the application generates an audit log once the user commits a transaction. To implement auditing, three events provided by nHibernate are used, namely IPostInsertEventListener, IPostUpdateEventListener, and IPostDeleteEventListener. IPostInsertEventListener event is triggered once a transient entity is persisted for the first time.

Each class that requires auditing is marked with *Auditable* attribute, which is then used to create audit logs for classes containing this attribute. All mapped properties are then audited by default and a suppress audit attribute is added to suppress auditing of a target property. Usually, and by default, all properties are audited. However, in special cases where auditing is not required, the *SuppressAudit* attribute is added to the property. In Algorithm 1, we show the process of generating the audit log when *IPostInsertEventListener6* event is triggered.

When an audit entry is created, it contains a session ID (transaction ID), a class name, an event type (Insert,
Update, or Delete), audit ID, creation date, user ID, URL, and a collection of values for all properties. The collection of values consists of the old value before the update and the new value resulting from the update. Moreover, during an update, old and new values are compared. Only if the two values are different from one another, the change is committed to the audit log. In Algorithm 2, we outline this procedure of updating audit logs. Currently, these audit logs are saved inside an RDBMS using two tables, the *AuditLog* table, and the *AuditLogDetail* table. Furthermore, Globally Unique Identifiers(GUID) are used as primary keys in auditlog tables.

Once a change is observed in a class, the ORM's event handler is invoked. Similarly, the event handler is also invoked when the change is observed in the "AuditLog" and the "AuditLogDetail" classes. Lines 2–5 in Algorithm 1 and Algorithm 2, prevent the creation of logs for Audit Classes. In the absence of this condition, the event logger would fall into an infinite event loop.The infinite loop can also be prevented by removing the "AuditableAttribute" from the audit classes. However, we use lines 2–5 as a check to avoid the loop in case a developer adds the attribute by mistake.

Once an audit log is generated, the application provides a link to the audit log page from the primary object. The link allows end users to look at the object history and track any discrepancy caused by a bug or malicious activity.

Blockchain Integration to Audit Logs

In this section, we will show how audit logs, obtained from our application, are integrated with the blockchain. So far in our design, we have an application that stores audit logs upon receiving a transaction. Now, we need to convert the audit log data into a blockchain-compatible format (blockchain transactions) and construct a distributed peer-to-peer network to replicate the state of the blockchain over multiple nodes. In our current implementation the audit log is generated using the ORM, which calls a Representational State Transfer(REST)
Application Programming Interface(API) to store the audit log entry.

We used the ORM to create audit logs because the ORM acts as the gateway to capture all database transactions. Therefore, it is efficient to take advantage of ORM events to capture all the database changes and convert them into a JSON packet for the REST API. Our design is flexible and generic, and can also be used by other applications that do not use the ORM. Other than the ORM, the application layer or the data access layer can also be extended to capture the database changes in a JSON format and invoke the REST API. Moreover, the REST API can also be used by applications built using a serverless architecture.

5.3.1 Creating Blockchain Network. In *BlockAudit*, the network consists of peers that all have the privilege of accessing the application and creating an audit log. This network is connected in peer-to-peer model [93] and each peer can connect to all the other peers in the network. Connecting to a bigger subset of peers is beneficial, because it can avoid unnecessary delays in receiving critical information.

Algorithm 1: Creation of the audit log entry for persisting new objects to the database

```
Function OnPostInsert(PostInsertEvent e)
    if (e.Entity = AuditLog) then
        return;
                ▷ Do not create log entry for AuditLog
    if (e.Entity = AuditLogDetail) then
        return;
        ▷ Do not create log entry for AuditLogDetail
    if e.HasAttribute(AuditableAttribute) then
        var new AuditLog(SessionId,
          AuditEventType.INSERT,
          EntityName, EntityId, UserId, Url);
    for i = 0; i <
      e.Persister.PropertyNames.Length − 1 do
        if (suppressedProp.Contains(propName))
          then
            continue;
        auditLog.AddDetail(propName, oldValue,
          newValue);
    if (auditLog.Details.Any() then
        SaveToBlockchain(auditLog);
```

Algorithm 2: Creation of the audit log entry for persisting existing objects to the database

```
Function OnPostUpdate(PostUpdateEvent e)
                ▷ Do not create log entry for AuditLog
    if (e.Entity = AuditLog) then
        return;
        ▷ Do not create log entry for AuditLogDetail
    if (e.Entity = AuditLogDetail) then
        return;
    if e.HasAttribute(AuditableAttribute) then
        var new AuditLog(SessionId,
          AuditEventType.UPDATE,
          EntityName, EntityId, UserId, Url);
    for
      i = 0; i < e.Persister.PropNames.Length − 1
      do
        if (suppressedProp.Contains(propName))
          then
            continue;
        if (oldldValue <>newValue)) then
            auditLog.AddDetail(propName,
              oldValue, newValue);
    if (auditLog.Details.Any() then
        SaveToBlockchain(auditLog);
```

Table 2. Clear Village's actual transaction sizes in (bytes) for the three transaction schemes.

Type	Max	Min	Average
Per Transaction	501, 760	321	9, 302.81
Per Record	31, 104	319	2, 617.80
Fixed Length	32	32	32.00

Table 3. The description of fields of audit log JSON packet.

Field Name		Description
AppId		Unique identifier for the application
ClassName		The name of updated application class
CreatedDate		Creation date and time for the audit record
EntityId		Unique key for business object
EventType		This would be Update, Insert or Delete
Id		Unique id of the audit record, GUID
SessionId		Unique Id for a transaction
Url		Application page creating the audit
UserId		User id, for the user making the change
Details $(0-n)$	**Id**	Unique Id for the detail record
	NewValue	Current property value
	OldValue	Old Property value
	Name	Name of property e.g owner's name

Access Control. As mentioned in previous section, access controls may vary across blockchain application. These applications can be permissionless (open access) or permissioned (selective access). In permissionless applications, such as Bitcoin, an arbitrary user can download the Bitcoin Core software and join the network. However, in the private and permissioned blockchains, an access control mechanism is applied that restricts the participation to only approved users. Since audit logs consist of sensitive data, therefore, in BlockAudit we use a permissioned blockchain with access control provisioned to selected users. In permissioned blockchains, adjusting access control is trivial since any custom membership service can be used for the access control [94]. To avoid runtime complexities, we do peer screening prior to

network creation. The peer screening is done based on the IP addresses in which we curate a list of IP addresses, compile them in executable code, and provide the code to each peer. Upon executing the code, the peer gets connected to the network.

Additionally, each node is required to keep a copy of the blockchain at their servers and maintain a persistent connection with their corresponding application server. Persistent connections are necessary to maintain an up-to-date view of the blockchain in order to process, validate, and forward transactions, as well as to avoid unwanted forks and partitioning attacks that may result from an outdated blockchain view.

Listing 1. Blockchain transaction generated after serializing data from the audit log. This transaction is exchanged among the peers during the application runtime.

```
{  "AppId":"USA-FL-0000005",
   "ClassName":"SAGE.BL.InspSystem.PermitInspection",
   "CreatedDate":"\/Date(1532366360155-0400)\/",
   "EntityId":161031,
   "EventType":UPDATE,
   "Id":"9ceb8c2c-154a-49d5-9441-a92600db997b",
   "SessionId":"c66207c8-63be-4703-b858-cbfae98a988e",
   "Url":"\/SAGE\/Building\/Inspection\/InspectionReport.aspx?srcTp=309&srcId=17552018&
       InspectionTypeId=61663",
   "UserId":666,
   "Details":[
      {
         "Id":"fa268eaf-7993-48e3-ae6a-a92600db997b",
         "NewValue":"10",
         "OldValue":"9",
         "PropertyName":"DBVersion"
      },
      {
         "Id":"ee2cdbc2-9c3a-4bc9-afba-a92600db997b",
         "NewValue":"available after 1:00 pm",
         "OldValue":"available after 2:00 pm",
         "PropertyName":"RequestComments"
      }
   ]
}
```

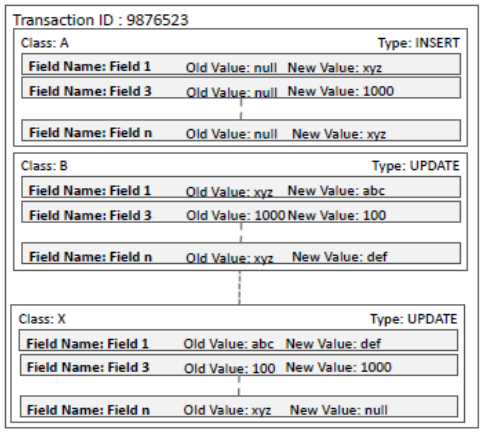


Figure 22: Audit generation for a transaction spanning across multiple objects.

5.3.2 Creating Blockchain Transactions. Once the network architecture is laid out, the next

step is to create blockchain-compatible transactions from the audit log data. For that, we convert the audit log data to a JavaScript Object Notation(JSON) format [95]. We preferred JSON over other standard data storage formats such as XML, due to its data structure compactness and storage flexibility. To obtain a blockchain transaction, we first pass the audit log data to a function that serializes it to JSON and calls *createAudit* REST [96] web service to create the audit log transaction. Each JSON packet is then treated as a blockchain transaction, and as soon as a node in the network receives a transaction, it broadcasts the packet to the rest of the network. Nodes can connect to multiple peers to avoid the risk of delayed transactions due to malicious peer behavior or network latency. In Table 2, we show the average transaction size from our sample system for October 2018. In Table 3 we describe the purpose network receives a transaction, it broadcasts the packet to the rest of the network. Nodes can connect to multiple peers to avoid the risk of delayed transactions due to malicious peer behavior or network latency. In Table 2, we show the average transaction size from our sample system for October 2018. In Table 3 we describe the purpose

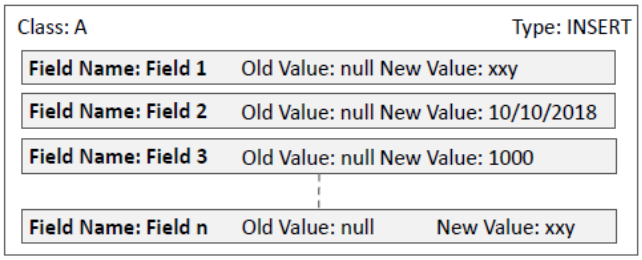


Figure 23: Audit Entry generation for an object.

of each field in the audit JSON packet and in Listing 1, we show the data structure of the blockchain transaction that is obtained after serializing data from the audit log.

Log for table/class. The audit event logger can also create a packet for each object in a transaction. We used this method in the prior work [104] and found that the packet size was small, however, the number of web service calls for each application transaction was high. For instance, if a transaction contains 10 classes, it will create 10 web service calls. While 10 calls can be handled by ORM-based audit logs, they are not optimal for blockchain-based audit logs. Log for transaction. The audit event logger creates a packet containing all insertions, updates, and deletions, that span across one or more objects, and sends the packet to BlockAudit as shown in Figure 22. Since the audit log data is consolidated, therefore, it is hard to search for updates for a specific class, which is a typical use case. Creating an audit log for a transaction reduces the number of web service calls and improves efficiency, and this design is more suited to blockchain based audit logs.

Consensus Protocol

The next phase in the BlockAudit design is the use of a consensus scheme among the peers to develop their agreement over the sequence of transactions and the state of the blockchain. There are various consensus algorithms used in blockchains, such as proof-of-work (PoW), proof-of-stake (PoS), proof-of-knowledge (PoK), Byzantine fault tolerance (BFT), etc. [97-98].

In Table 3, we compare the popular blockchain consensus algorithms. Notice that PoW and PoS have high scalability and fault tolerance. More specifically, they can scale beyond 10,000 nodes and can tolerate up to 50% malicious replicas. On the downside, they have low throughput and high confirmation time [99-100]. In contrast, PBFT has high throughput and low confirmation time. However, PBFT has low fault tolerance which makes it less suitable for permisionless settings.

For *BlockAudit*, we use PBFT consensus algorithm [101-102], which was originally designed to facilitate the decision-making process in a distributed environment. *BlockAudit* uses a permissioned blockchain system [103], in which all network participants are known to one another, and there is a weaker notion of anonymity. Since our system is primarily a private and permissioned blockchain, therefore, we are not constrained by high scalability challenges. Although in the future, we aim to extend our design to a bigger network, however, at the prototype stage, we are less than 100 peers. Due to high throughput and low latency, naturally, PBFT is more suited for our design.

In PBFT, the system comprises of a client that issues a request (transaction), and a group of replicas that execute the request. The primary replica orders transactions and relays them to other replicas. The transaction is processed in four stages, namely pre-prepare, prepare, commit, and reply. When the client receives a minimum of $3f + 1$ responses, f being the number of faulty replicas, the transaction processed. In Figure 7, we provide an illustration of PBFT, which we later use to design and calibrate *BlockAudit*. In Figure 8, we show the complete design of *BlockAudit*, where the blockchain is integrated with the serialized JSON output of the business application.

Table 4. An overview of popular consensus algorithms used in blockchains.

Properties	PoW	PoS	PBFT
Blockchain Type	Permisssionless	Permissionless	Permissioned
Participation Cost	Yes	Yes	No
Trust Model	Untrusted	Untrusted	Semi-trusted
Scalability	High	High	Low
Throughput	<10	<1,000	<10,000
Byzantine Fault Tolerance	50%	50%	33%
Crash Fault Tolerance	50%	50%	33%
Confirmation Time	>100s	<100s	<10s

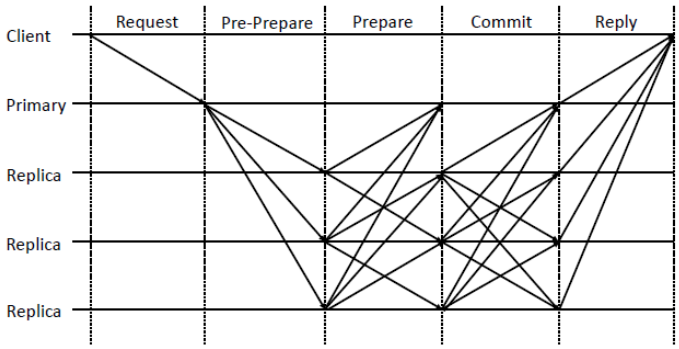


Figure 24: An overview of PBFT protocol with client issues a request to the primary replica.

F. Blockaudit Analysis

In this section, we analyze various aspects of *BlockAudit*, including design, complexity, and security analysis.

A) Design Analysis

In *BlockAudit*, each peer uses the ORM-based audit log application that is connected to a database. Once the ORM observes a change, it updates the database and issues a transaction, and sends it to the primary replica. The primary orders the transaction and broadcasts them to all the other replicas. Upon receiving the transaction, each replica checks if the transaction is valid and follows the correct order. The order of the transaction is ensured by the timestamp, and the ordering rule involves the chronological sequencing of each transaction. In *BlockAudit*, the primary preforms transaction sequencing based on the time at which it receives transactions from the application replica. We use this approach as a security design choice to prevent malicious replicas from arbitrarily modifying their transaction timestamps. In the following, we show how transaction sequencing is performed in *BlockAudit*:

(1) An application generates a transaction at time t_i and the primary receives the transaction at time t_j.
(2) First, the primary checks if the transaction respects the temporal ordering ($i < j, \forall i, j$). This assumption is valid for any real-world system, since each transaction experiences a non-zero delay during transmission.
(3) If the primary observes a violation i.e., $> j$, it assumes that the application replica is misbehaving. Therefore, the primary discards the transaction.
(4) In the transaction confirmation phase, the active replicas also compare the time at which they receive a transaction to the time of the transaction generation. This serves as an additional security measure to ensure that the policy precedence is respected, even when ignored by the primary.

In *BlockAudit*, we enforce the ordering of transaction since it is critical in audit log applications. For instance, consider a case in which tx_a involves a change made to a class. The next transaction tx_b reverses the change made by tx_a, then it is critical to process before tx_b. Otherwise, the order will be violated and the audit log will reflect a different state of the

database than the actual.

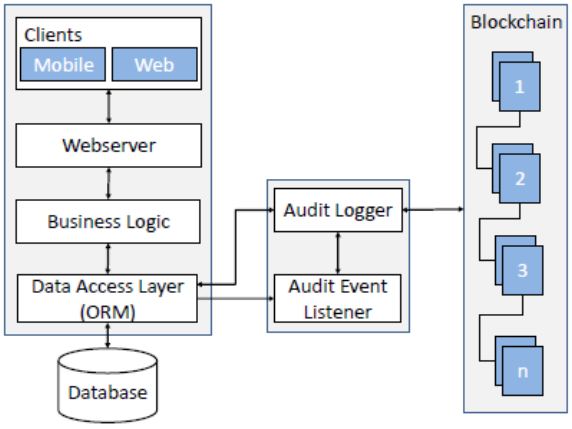


Figure 25: Complete system architecture of *BlockAudit* after blockchain is integrated to the JSON output.

In summary, *BlockAudit* constitutes of a client (audit log application) that generates blockchain compatible transaction, a primary replica that receives and orders transactions, and a group of active replicas that execute PBFT to generate a blockchain-based audit log. In conventional PBFT, the client is independent of the active replicas that execute the consensus protocol. In *BlockAudit*, the client is one of the active replicas that issues the transaction. In the verification process, the issuer becomes the client and all other replicas act as validators.

Key Takeaways. From the design implementation, we had the following takeaways: (1) PBFT-based permissioned blockchains are more suitable for audit log applications. (2) Extending ORM provides an efficient mechanism of converting database transaction to blockchain compatible transactions. (3) Existing application can seamlessly integrate with blockchain based audit logs using ORM extension. (4) REST based web services can also be easily extended to support applications that do not use ORM. (5) JSON format is the de facto standard for REST API's, and therefore efficient and suitable for an audit log transaction.

B) Complexity Analysis

A key aspect of PBFT-based blockchain systems is the time and space complexity associated with the network and the blockchain size. The time complexity partakes the time taken by replicas to develop consensus on a transaction or a block. The space complexity involves the storage and the search overhead that compounds due to append-only distributed blockchain design. In the following, we analyze these aspects of complexity in BlockAudit.

Time Complexity. To achieve consensus over the state of blockchain with n replicas, $n2 - n$ messages are exchanged, as shown in Figure 7. Therefore, for each transaction generated within the system, the overall complexity becomes $O(n2)$. Compared to PoW-based blockchains, in which the consensus complexity is $O(n)$, PBFT has a high message complexity which can lead to system overheads and delays. However, we argue that in PoW-based blockchains systems such as Bitcoin, the total number of active nodes are over 6-8k [63]. In

comparison, BlockAudit constitutes less than 100 peers. Therefore, it can tolerate this complexity overhead, keeping in view the other benefits associated with PBFT such as high throughput.

Space Complexity. The space complexity of the system can be ascribed to the overhead associated with the storage of blockchains at each peer. One major limitation of replacing the client-server model with a peer-to-peer blockchains system is that each peer is required to maintain a copy of blockchain. This leads to a high storage footprint since blockchains are always growing in size. The size footprint also increases the search complexity for transaction verification. For instance, when a newly generated transaction is sent to a group of peers for verification, they validate its authenticity by consulting its history in the blockchain. If the blockchain size is large, the verification time increases. As such, if the rate of the incoming transaction is high, then high verification time may lead to processing overhead, thereby increasing latency and reducing the throughput. In *BlockAudit*, the space complexity of a system, complementary to any other blockchain system is $O(n)$.

Key Takeaways. From the complexity analysis, we had the following takeaways: (1) PBFT-based based blockchains have high message complexity. Therefore, if the network scales beyond a few hundred nodes, the application may become inefficient. Therefore, we observe a tradeoff between the message complexity and the network scalability. (2) Generally, the space complexity of blockchain is high, due to the append-only model. In *BlockAudit*, the space complexity is similar to any other blockchain application.

C) Security analysis

An essential component of our work is the defense against the attacks outlined in the threat model §2.2. In this section, we discuss how *BlockAudit* defends against the physical access attack and the remote vulnerability attack. **Physical Access Attack**. In the physical access attack if the attacker acquires the credentials of a user, he can make changes to the application data using the application interface. In this case, his activity will be logged in *BlockAudit*. Since the log is kept in the blockchain by the user, the attacker will not be able to remove the traces of his activity. Therefore, when the attacker's activity is exposed, auditors will be able to track the tampered records and take corrective measures to restore data to the correct state. Moreover, if the attacker is able to get write access to the database, he will be able to change data in different tables. Since the audit log generation is at the ORM level, therefore, these changes will not be present in the audit log. This will enable the auditors to detect malicious activity and take preventive actions.

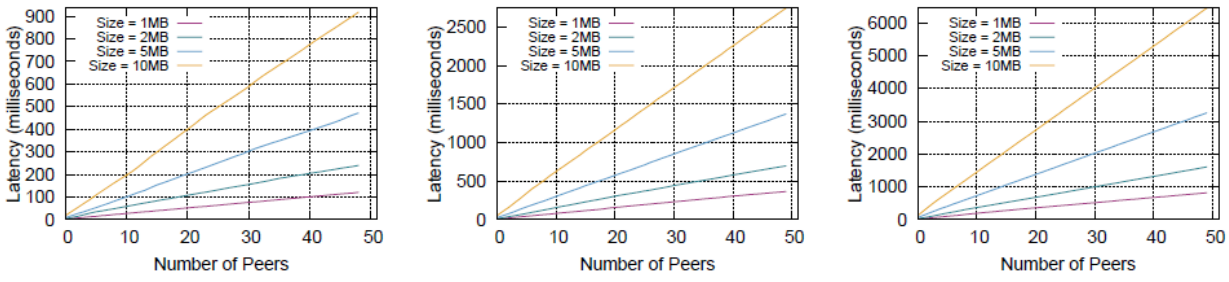
Remote Vulnerability Attack. In case of a remote vulnerability attack in which the attacker exploits a bug or vulnerability in the application, the audit log will show the effect of the changes or errors resulting from the attack. Additionally, the blockchain will also preserve the tamper-proof state of the audit log prior to the launch of the attack. As a result, the auditor will be able to compare the audit log and the current data to detect changes made during the attack. In the absence of the blockchain, if the attacker corrupts the prior state of the audit log, there is no way auditors can recover from it. However, with *BlockAudit*, not only the attacks are detected, but the system state is also recovered. Furthermore, for a successful attack in the presence of *BlockAudit*, the attacker will need to corrupt the blockchain maintained by each

node. Based on the design constructs and security guarantees of blockchains, corrupting blockchain repositories of a majority of nodes is costly, and therefore infeasible.

After realizing that *BlockAudit* is able to defend against the attacks outlined in our threat model §2.2, there are however few considerations to be made while using PBFT-based blockchain model. The prior work in this direction does not consider Byzantine behavior among nodes. In *BlockAudit*, we consider that peers may behave arbitrarily and create confusion in the view of other honest peers. Therefore, we want *BlockAudit* to be robust against malicious replicas. While other consensus mechanisms such as PoW may withstand up to 50% of faulty replicas in the system, PBFT, in contrast, has low fault tolerance. In a situation where there are f faulty replicas, a PBFT-based blockchain system needs to have $3f + 1$ honest replicas in order to function smoothly. Roughly speaking, PBFT-based blockchains require 70% nodes to behave honestly in order to avoid disagreements. However, in *BlockAudit*, we try to raise the threshold of fault tolerance by making minor adjustments to the security design.

Increasing Fault Tolerance. In a situation where there are r honest replicas in a blockchain and the attacker is able to position f faulty replicas such that $4f + 1 > f + r$, then the attacker will be able to stop transaction verification and may even cause forks. To counter this, we propose an expected verification time window W_t which will be set by the primary replica before passing the transaction to the verifying replicas. The primary replica knows the total number of active replicas in the system and can calculate the total number of messages to be exchanged until the transaction gets verified. In this case, the total number of messages will be in the order of $(f + r)2 - (f + r)$. Let $c \times t_b$ be the time taken for the transaction confirmation, where c is an arbitrary constant set by the primary replica. Based on these values, the primary replica can set an expected time window $W_t \geq c \times t_b$ in which it expects all peers to validate the transaction and submit their response. Let t_{start} be the start time at which the primary replica initiates the transaction. If by W_t the primary does not receive the expected number of responses from the replicas, it will abort the verification process and notify the auditor.

Depending on the application's sensitivity, the primary replica can either set another optimistic value of W'_t, where $W'_t \geq W_t$, and repeat the process or it can simply abort the process and notify the application auditors regarding the malicious activity. We leave that decision to the audit log application and its sensitivity to malicious activities.



(a) Transaction Rate λ =200 tx/s (b) Transaction Rate λ =3,000 tx/s (c) Transaction Rate λ =6,000 tx/s

Figure 26: Time taken to reach consensus at different types of audit transaction with varying transaction rate λ (200-6,000 tx/second).

However, in our experiments, we relax the condition of sensitivity and re-submit the transaction for another round of verification. We set a new expected verification time window W'_t and wait for the response. Our choice of relaxing the condition of sensitivity is owing to the unexpected delays in the message propagation; given that our system would run over the Internet. However, if the primary replica does not receive the approval for the second time, it aborts the process and notifies the application.

Detecting Malicious Nodes. In BlockAudit, we also enable detection of the malicious nodes that corrupt the process of transaction verification. For that, we store the identity of the replica in each iteration of the response. For instance, in the first iteration of W_t, we note the identity of replicas that send their digitally signed approval for the transaction. Let h be the subset of replicas that send their response in the first iteration, where $h \leq (f + r)$. The primary replica stores the identities of replicas in h and initiates the second iteration at t'_{start} and waits for response till W'_t. Upon receiving the response in the second iteration, the primary replica updates h and removes the duplicates. By comparing h with the identity of all the replicas, the primary replica can find the malicious replicas and request their removal from the verification process.

It is possible that an adversary, aware of the two-phased approval process, may attempt to trick the system by sending a response from a subset of malicious peers in each phase of approval. For instance, the adversary can split his set of malicious replicas in f_1 and f_2, where $f_1 + f_2 = f$. In the first phase of approval, the adversary can send a response from f_1 replicas. However, the adversary ensures that $3f_1 + 1 \geq r$, so that the transaction does not get enough approvals to be accepted by the primary replica. The primary replica will append f_1 to its set of h. In the second iteration, the adversary will incorporate signatures from f_2, and the primary replica will also add them to h. As a result, the primary replica will not be able to detect the actual number of malicious replicas in the system.

To counter that, we randomize the two-phase approval process to v-phase approval process, v may take any value of the primary replica's choice. When the transaction fails the first attempt, the primary replica can either abort or continue the approval process. Continuing from the above-outlined scenario, if $v = 3$, then the attacker will either have to include one of f_1 or f_2 replicas in the third phase. And if the primary replica iterates one more time, the adversary will be bounded to include the set of replicas that he did not include in the previous iteration. As such, the primary replica will notice the incoherence in the response of a few replicas in each iteration of the approval process, and the adversary will risk exposing his malicious replicas. Although this procedure ensures high security and the ultimate exposure of adversary in the process of verification, it is, however, time-consuming and may lead to a transaction stall. Again, we leave this to the primary replica, which can make decisions that suit the application requirements.

Key Takeaways. From the security analysis, we had the following takeaways: (1) BlockAudit counters the con-ventional audit log attacks namely the physical access attack and the remote vulnerability attack. (2) Additionally, BlockAudit also makes audit logs secure against Byzantine behavior, tolerating up to 30% malicious replicas. (3) Leveraging the design policies

in permission settings, BlockAudit is able to detect malicious replicas.

G. Experiment and Evaluation

In this section, we present experiments carried out to evaluate the performance of *BlockAudit*. First, we extended the nHibernate ORM to generate a serialized JSON output in the form of transactions as shown in Listing 1. The transactions are broadcast to the network where a *BlockAudit* blockchain instance is configured at each node. For experiments, we used sockets to set up the network and a NodeJS client to receive JSON transactions.

Simulation Environment. We simulated our blockchain network using a LAN setup at our research lab. We used 20 machines, each running the Linux OS with Intel Core i5 processor and a 16MB RAM. Next, we set up a virtual environment at each node to construct a multi-host network. We assigned port numbers and sockets to each host that acted as a peer. The socket connections were used to exchange data with peers using IP addresses and port numbers. Each peer was equipped with a JSON master list that contained the information of all the other nodes. Data packets of the desired size were generated and broadcast over the network. We encoded the PBFT protocol in NodeJS and executed it over all the peers. The selection of the primary replica can be done using any method suitable for the application. In BlockAudit, in each iteration, we selected the primary in Round-robin manner. To reflect the real-world delays in our simulation, we manually added a round-trip delay of 100ms in each transaction broadcast over the network. Finally, once the transaction obtained sufficient approvals, it was added to the blockchain of the primary replica, and subsequently, all the other replicas.

We evaluate the performance of our system by measuring the latency over the consensus achieved by peers. We increase the transaction payload size from 2MB to 20MB and the rate of transaction λ from 200 transactions per second to 6,000 transactions per second. By adjusting these parameters, we monitor the time taken by peers to approve the transaction. Let t_g be the transaction generation time, and t_c be the time at which it gets approval from all active peers. In that case, the latency l_t is calculated as the difference between t_c and $t_g = t_c - t_g$, where $t_c > t_g$. We report the simulation results in Figure 26.

Simulation Results. Our results show that irrespective of the payload size, the latency margins remain negligible as long as the number of peers is less than 30. As the size of the network grows beyond 30 nodes, the latency factor increases considerably. Furthermore, we also notice that a sharp increase in latency when the payload size changes from 5–10MB and a negligible change in latency when the payload size changes from 15–20MB.

We also noticed that as the rate of transaction λ increases from 200 transactions per second to 6,000 transactions per second, the confirmation time for transaction also increases. Intuitively, this can be attributed to the processing overhead caused by the increasing rate of λ at each replica. However, it can be observed from 9(c) that within a network size of 50 peers, *BlockAudit* has the capability of processing 1,000 transactions per second, with the payload size of 10 MB. This payload size is equivalent to 10 blocks in Bitcoin. For the payload size of 1MB, *BlockAudit* achieves a throughput of 6,000 transactions per second. Considering low throughput of conventional blockchains (3–7 transactions/second in Bitcoin), *BlockAudit*

achieves high throughput. This also justifies our choice of using PBFT as consensus scheme for our system.

Evaluation parameters obtained from our experiments can be used to define the block size and the network size, specific to the needs of the application. As part of our future work, we will use these parameters along with other consensus schemes to find optimum block size and the average block time for the audit log application. By varying consensus schemes, we will be able to compare and contrast the performance of various design choices and select the best that can be used for *BlockAudit*.

H. **Discussion and Future Work**

With *BlockAudit*, we were able to meet our overall objective of securing audit logs using blockchains. We show with theoretical analysis and simulations that our system is secure and efficient, and it achieves high throughput(§7) by using the PBFT consensus protocol. In *BlockAudit*, audit log transactions were seamlessly generated with minor changes to the existing system. Moreover, *BlockAudit* can be plugged into any enterprise business application, that consumes a REST API to send audit log data as a transaction. In summary, we successfully extended our application into the blockchain paradigm to harden its security and increases the overall trust in the application. Our system is robust against the physical access attack and the remote vulnerability attack.

Limitations. Despite all the promising outcomes, there are, however, two major limitations in *BlockAudit*. The first constraint is the high message complexity due to PBFT, and the second is a high storage footprint due to data redundancy in the blockchain design. Since in PBFT, the message complexity is high ($O(n2)$), therefore, in adverse network conditions, PBFT may perform poorly, compared to other consensus protocol [105]. In spite of these limitations *BlockAudit* performs within the requirements of our application, and could support PayPal [105] which processes 170 transactions/second, however, our solution would not be feasible for Visa which has a transaction rate of 2000 transactions/second [107]. Secondly, audit logs by design have a high storage footprint, as each transaction in the system has a corresponding entry in the audit logs. In *BlockAudit*, the problem is further increased since transactions are replicated on multiple peers, resulting in high storage overhead.

Keeping in view these limitations, we propose that high message complexity can be resolved by using other newly proposed consensus algorithms such as Clique [108], that belongs to the family of Proof-of-Authority consensus protocols. Clique has a message complexity of $O(n)$, which is considerably lower than PBFT and PoW. Using Clique may allow us to support a larger number of peers, achieve high throughput, and reduce confirmation delays of transactions in *BlockAudit*. However, in Clique, peers run into the risk of multiple views at the same time. In blockchains, this inconsistency is called a blockchain fork. These forks can lead to temporary or permanent partitioning in the network. Currently, we are exploring methods of fork resolution in Clique, and therefore applying it in *BlockAudit* is part of our future work.

The space complexity can be reduced by adding data retention policy and purging data after

its fixed retention time. This would optimize the overall size of the blockchain, and lead to less storage and search complexity. In addition to these two schemes, we also propose two other optimization strategies to meet the design limitations in *BlockAudit*.

Another limitation in *BlockAudit* is the weak link between the application and the audit log. In the current implementation, if the application itself is compromised, and subsequently the audit log generation fails, then *BlockAudit* will not be able to detect the fault at the application. At present, *BlockAudit* enables applications to seamlessly integrate with blockchain system and benefit from it. Therefore,*BlockAudit* remains agnostic to the application itself and the data being produced by it. As a result, we observe a trade off between the seamless integration of audit logs with the application and the enhanced security of the audit log generation interface. Currently, *BlockAudit* is designed to facilitate the integration of audit logs with eGovernment application. In future, we also aim to focus on detection application-level faults in *BlockAudit*.

The latency is a critical problem in distributed systems, which can be 1) latency due to the consensus scheme operation, and 2) latency due to network conditions. To minimize latency due to consensus, we select consensus algorithms, such as PBFT, which is known to provide low latency and high throughput compared to other popular schemes such as PoW. We note that such a choice comes at a certain cost: PoW is known to have better security, since it tolerates up to 50% Byzantine nodes while PBFT tolerates only 30% [109]. Acknowledging that, and giving latency a higher priority over security, in *BlockAudit*, we made the consensus choice to minimize the latency.

The other component of latency is due to the network, which includes transmission and propagation delays under a certain payload size. In *BlockAudit*, and as shown in Figure 26, with a payload of 10MB and a network of 50 replicas, the transaction confirmation experiences a delay of 6 seconds. In *BlockAudit*, this is an upper bound on the end-to-end latency, which is considerably low compared to 600 seconds of delay in Bitcoin. For our Enterprise application, this delay is tolerable. However, if *BlockAudit* is to be extended for applications with larger payloads, we suggest two improvements as the latency increases. First, the communication medium between applications can be enhanced to support high bandwidth. Second, localities could be exploited to host applications within the same autonomous system to reduce propagation delays. Implementing these improvements is a future work.

Optimization. To increase the performance and to keep the audit log tamper-proof, we propose having two sets of blockchains, namely the recovery blockchain, and the detection blockchain. In Figure 27 we provide a system overview of this two blockchain system. The recovery blockchain stores the complete audit log transaction, including details of all data changes in an application-level transaction. The recovery blockchain can be used to restore data to its prior state, which would be the state of data before an attack. The recovery blockchain would

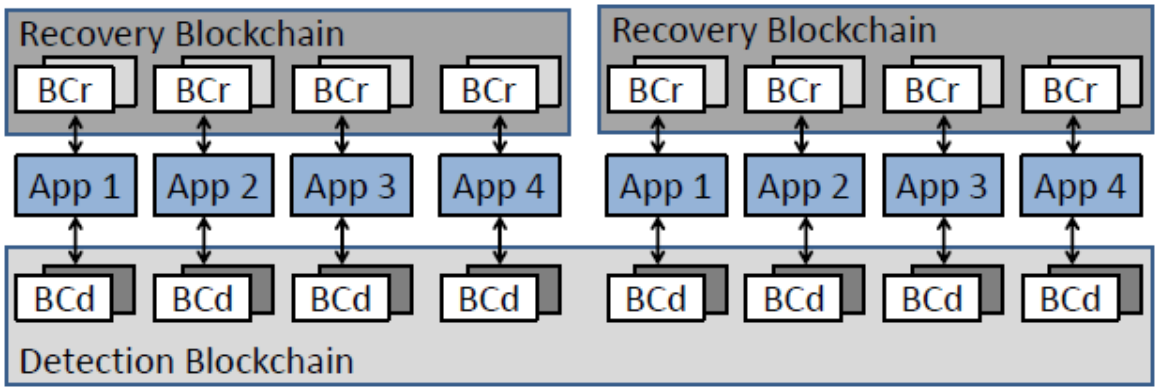


Figure 27: Audit log block chain detection vs recovery.

require more space, and longer consensus time due to large transaction audit data packets. The number of peers k in the recovery blockchain can be kept small to ensure immediate consensus and avoid delays. Since the security of PBFT relies on the faithful execution of the protocol by at least 70% replicas, therefore for a baseline, the minimum size of the recovery blockchain must be four nodes considering one malicious replica ($k \geq 4$).

The detection blockchain can be used to detect audit log tampering only. It will not have the information to recover the audit log to a correct state before the attack. The business application will generate a cryptographic hash using the audit transaction. The hash and a unique transaction identifier will be stored in the blockchain. In the case of data tampering in the audit log, the newly computed hash will not match the hash stored in an audit log. This will indicate that the audit log has tampered. Once tampering is detected, application administrators could use corrective measures to fix the security breach. Atop that, data can could be restored to the previous state by using database backups. The recovery blockchain which will have K_d peers, where $K_d > 2K$. Therefore, the adversary will have to compromise twice as many nodes to tamper the system without being detected. This optimization increases security and provides a second layer of defense.

Despite the existing challenges, *BlockAudit* is a feasible approach towards blockchain-based secure audit logs. Extending the capabilities of the prior work, *BlockAudit* brings the theoretical foundations into practice and as shown in section 7, it has been deployed and instrumented in a real blockchain network. Moreover, *BlockAudit* is also capable of ensuring operational consistency even in the presence of Byzantine replicas. Therefore, it is a better candidate for the audit log security and can be applied to eGovernment solutions.

I. Conclusion

We present a blockchain-based audit log system called *BlockAudit*, that leverages the security features of blockchain technology to create distributed, append-only, and tamper-proof audit logs. We highlight the security vulnerabilities in existing audit log applications and propose a new design that extends NHibernate ORM to create blockchain-driven audit logs. For our experiment, we used an application provided by ClearVillage inc to generate transactions from

audit logs, and record them in our custom built blockchain. By design, *BlockAudit* is agile, plug and play, and secure against internal and external attacks. In the future, we will extend the capabilities of *BlockAudit* by deploying it in a production environment and explore various performance bottlenecks and optimization techniques.

3.5 Architecting Blockchain for IoBT and its Performance Evaluation

Military technology is ever-evolving to increase the safety and security of soldiers on the battlefield through integration of IoT solutions. This aims to improve the operational efficiency of mission-oriented tasks on the battlefield significantly. The state-of-art battlefield networks are traditionally reliant on centralized communication paradigms, which imposes security and safety concerns when the battlefield things scale up. Since the centralized architectures suffer from security issues such as denial of service attacks, and central point of failures, a flexible architecture that is mobile, resilient, and adaptable for dynamic topologies where the number of nodes can be unpredictable must be designed. Introducing decentralized platforms for managing Internet of Battlefield Things (IoBT) could potentially address previous-mentioned security issues and safety concerns of traditional battlefield networks. Blockchain, a decentralized customizable platform, can be a useful tool to securely interconnect multiple nodes in an IoBT environment and securely offer mission-aligned operational capabilities in the battlefield using the distributed ledger. Such IoBT nodes can be interconnected in a peer-to-peer network that maintains a distributed ledger in a flexible and dynamic topology. In this section, we integrate blockchain in the IoBT context to propose a 3-layered architecture and evaluate its performance with the goal of determining its potential to serve required performance and security needs of IoBT environment. Using different testing parameters with dynamic topologies, the metric data would help in suggesting the best parameter set, network configuration, topology, and blockchain usability views in IoBT. The implementation of a dynamic and asynchronous topology would help to categorize the frequency and update behavior of the distributed ledger. We show that a blockchain integrated IoBT platform has heavy dependency on the characteristics of the underlying network such as topology, link bandwidth, jitter, number of nodes, nodes' availability, and other communication configurations. A wide range of values for these reconfigurable parameters can be used in practical scenarios to find if the simulated values can meet the IoBT performance needs and how can we tune them up to achieve optimal performance.

3.5.1 Motivation

Internet of Things (IoT) technologies provide consumers with smart devices and sensors that are capable of delivering control and information across the Web. Due to their small scale, these systems are constrained in terms of performance, causing oversight when it comes to security. For this reason, exploitation of these devices is common and trivial and the result is usually catastrophic. The Mirai botnet [110] is one of many malware that has infected millions of devices. In particular, Mirai targeted active telnet services (running inadvertently), by exploiting weak authentication and default credentials. Devices were compromised and weaponized to create Distributed Denial of Service (DDoS) attacks that crippled Internet services across the globe. Compromises on these devices is commonplace due to the lack of vetted security mechanisms such as those that are found in enterprise networks. Strong encryption, rigorous security testing, and relevant vulnerability knowledge bases, are lacking on these small scale

devices. In the military world, soldiers are starting to rely on IoT devices for situational awareness, navigation, and others to make critical decisions for mission success. In the tactical environment, these devices and systems must ensure confidentiality, integrity, and availability in battlefield conditions. Adverse location, limited power, and low bandwidth in decentralized and mobile topologies are among many of the conditions. Securing communications across IoT and IoBT systems is difficult, but critical for the warfighter.

The traditional IoBT networks are centralized and thus vulnerable to a single point of failure attacks, i.e., if the central server gets compromised, all data and operations in the network are affected [111]. In addition to the architectural demerit, the security issues on these devices and networks are mostly overlooked due to lack of computing capabilities. Therefore, majority of applications focus on the mission-related performance of the device instead of security [112, 115]. As the next generation IoBT networks are becoming more decentralized, it important to adopt a decentralized solution to eliminate the malicious activities involved in the IoBT infrastructure. The emerging blockchain technology offers a tamper-resistant distributed ledger platform that can potentially be leveraged to track mission related activities in the battlefield. Although the blockchain technology was introduced to design cryptocurrencies, its fundamental properties can be successfully applied to networking and communication fields. Blockchain leverages a distributed network where each peer maintains a local copy of a distributed ledger that is immutable. Blockchain technology can be a useful tool because it can assure the integrity and accountability of all activities through validating transactions [113] by a group of nodes instead of single administrator. Since blockchain relies on cryptographic primitives, special considerations must be taken before its implementation in IoBT networks [114, 116]. The state-of-art blockchain platform suffers from several performance challenges such as scalability issues in terms of both devices and transactions, latency to validate transactions, and extensive network overhead required to maintain peer communication. Significant consideration must be taken to address the impact on transactions throughput and bandwidth overhead; such attributes are necessary for the dynamic and constrained networks such as military environments.

3.5.2 Next Generation Battlefield Characteristics

The military technology is rapidly evolving as the IoT devices and sensor technology is providing a major boost in the consumer industry. However, wide adoption of IoT technology also increases the attack surface of the integrated platform to disrupt the normal activities and introduce anomalies. There are many different components that make up a battlefield, from the human capital to the technology that soldiers carry on their person. Although there exist numerous security issues when considering heterogeneous elements in a battlefield, problems like network traffic, communication mediums, and node location are often targeted frequently due to their wide vector of attack. Before we dig deeper into a blockchain- based solution, we must first explore some of the fundamental reasons for a new system.

Attack space: A battlefield consists of a combination of devices using different communication mediums to provide up-to-date intelligence on what is occurring on the inaccessible terrain. Soldiers need to rely on this information to make timely decisions that impact not only the next move, but potentially the outcome of the mission. The loss and corruption of this information could be catastrophic, resulting in loss of lives and valuable resources, along with breach to the

overall security. Attackers aim at stopping information flow by using different attacks, such as Denial of Service (DoS), data fabrication, spoofing, data manipulation and others as described in [117]. In many cases, these attacks are carried out by exploiting weaknesses in Internet protocols, information handling, or simply by volume influx. Mitigation for DoS attacks include intrusion systems, packet analysis or firewalls, however, the nodes operating in a battlefield may not have sufficient energy to use these techniques [118]. Thus, availing redundancy in the IoBT services is very important in order to avoid DoS type of attacks which could happen in the battlefield through jammers.

Communication: Home users have the convenience of using LTE, Wi-Fi, and Ethernet to remain consistently connected to web and Internet. The infrastructure in place allows users to move from one location to the next seamlessly without losing connectivity. For the military scenario, these mediums typically work for stationary bases or short range missions, but not in cases where soldiers may travel longer distances on uneven terrain. The current technology in place, such as mobile subscriber equipment, satellites, long-range radios, and others [119] allow for operations to continue, but may provide few tactical advantages in current-day missions due to evolved attacker technology. Army researchers have since invested into a different method of communication: networks and secure wireless [120]. Ad-hoc networks and true mobility allow for communications to continue, but they must be partnered with a secure system to function as required. These networks may still suffer from attacks like signal jamming, but offer high resiliency and geographic range.

Node location: A critical central component for any network is the topology being used. In most cases, a network setup will directly affect network traffic, consumption, throughput, and more. Additionally, the locations of nodes and routers on a network will directly affect the security of a system. When a router is the central point of communication between all nodes, a very specific target is identified for attackers. Centralized systems have shown flaws and security gaps, namely that of high dependency and single point of failure. Redundantly connected systems, which at their core are decentralized, allow for the mitigation of threats like data corruption, data loss and service interruptions. These issues may have independent solutions, though they may be costly. Take for example [121], where an analysis of different attack management platforms for DoS attacks is presented. Although the solutions are many, they requires a memory intensive solution that may not be viable in all environments. For communication security, defenses such as [122] involve adding more nodes into a network that may be used as an early warning system. Since centralized networks suffer from bigger attack vectors that make it difficult to always mount a perfect defense, tactical communication architecture is preferred to be decentralized.

Rationale to design blockchain-based IoBT infrastructure:

The massive scale and distributed nature of IoBT devices will create several security and privacy challenges. Firstly, the underlying IoBT networking and communication infrastructure needs to be flexible and adaptive to support dynamic military missions. This dynamic change to the communication infrastructure needs to happen in an autonomous fashion without reliance on centralized maintenance services. Second, there is a need to ensure the veracity of the information made available through the IoBT devices. There is a need for building a trusted platform to ensure the information consumed by the human warfighters is accurate. Finally, the adversaries can compromise the IoBT devices to impact the mission negatively. There is a need

for a platform that can balance the tradeoff between resilience and risk of conducting operations in a decentralized fashion.

Therefore, we focus on developing a Blockchain empowered trusted architecture for IoBT. The architecture aims to address trust, privacy and security challenges in the IoBT. The architecture comprises of three layers: battlefield sensing layer, network layer and consensus and service layer. We provide the design of each of the layers and the accompanying research challenges. To realize the resilient Blockchain-enabled IoBT platform, the entities that have networking capability, such as ground vehicles, wearable gadgets, handheld weapons, satellites, unmanned systems, command & control (C2) base, and so on., need to be tightly coupled together for establishing communication links among each other. With a priority on the mission related operations, the physical devices are also essential to maintain the distributed ledger that tracks the required transactions in the battlefield. The characteristics of transactions may vary depending on the military applications, for example, secure logistics management through supply chain tracking, assembly-line provenance, smart C2 surveillance, mission planning and resource tracking, etc.

3.5.3 Background and Related Research

IoBT is a natural progression for Blockchain-based IoT technology. Just as home users can benefit from the data collection and monitoring of their home devices, military personnel can greatly benefit from sensors in the battlefield that provide better situational awareness to them and to the other military devices that might be acting autonomously. Yushi et al. [123] describes an early architecture view of a military internet of things (MIoT) system that relies heavily on server- client type of system, where sensors provide information to the networked servers. Farooq and and Zhu [124] studied the potential cost in power and resources to have a system that was not server-client based, but was more a Device-to-Device (D2D) type of interaction. The Peer-to-Peer (P2P) type of system can provide better robustness in the network because there are no central servers that are heavily relied upon by the system to operate successfully. The robustness of a P2P type of IoT system is tested by Abuzanab and Saad [125], where a game theory model was built and run to better understand how connectivity in a P2P system can be disrupted by attacks and how strategies to regain connection could maintain a connected and operational system

Overall, IoBT systems, like IoT systems, are still roughly centralized with a heavy dependence on management nodes. Abuzainab and Saad [125] and Farooq and Zhu [124] provide some insight to the benefits of a P2P system as well as the potential cost to keep a system of this type up and running. The added robustness that can be achieved in a P2P or D2D system appears to be a valuable point to consider in future IoBT systems because portions of the system are likely to be attacked. The challenges involved in more P2P IoBT are not really discussed and will require new protocols and rules that allow the nodes of the system to govern themselves without the management of central servers. It will also require new strategies to distribute the data shared among all the participating nodes in an efficient manner that would not suffer from an attack of a centralized data store.

The IoBT devices are limited in their ability to deploy sophisticated security mechanisms which leads to extremely exploitable IoBT networks. Researchers have investigated the security issues in IoBT. Specifically, the communication and information management challenges in an IoBT context have been studied [126]. Researchers have proposed an integrated IoT and network-centric warfare framework to address integrity issues in IoBT [127]. Researchers have proposed game theoretic approaches to optimize the interconnectivity in IoBT to support dynamic military missions [128]. However, these efforts do not address the need for a trusted platform for IoBT that can balance the tradeoff between resilience and risk from adversarial attacks. Given the past research on IoT is mostly focused on the centralized architecture standpoint, there is a strong need of decentralized framework for IoBT to serve the purpose of the battlefield environment.

3.5.4 Blockchain-enabled IoBT Architecture

Considering that the network-centric military operations can be audited and tracked using the distributed ledger system provided by Blockchain, this feature in fact improves the trust-worthiness of each action or command to execute in real-time. Our proposed architecture to achieve these goals constitutes the three important layers as shown in Figure 28. The bottom most layer is called as "Battlefield Sensing layer", where the sensor- equipped entities gather and disseminate information about the battlefield and collectively work toward achieving a common goal. The next upper level layer is the "Network layer", which is created considering a dynamic topology among a set of capable military nodes, that will be serving for Blockchain related operations. The top layer is the "Consensus and Service layer", which serves the purpose of defining individual roles and mechanisms to maintain Blockchain consistency. Each layer has different functionalities and responsibilities, which we briefly describe in the following subsections.

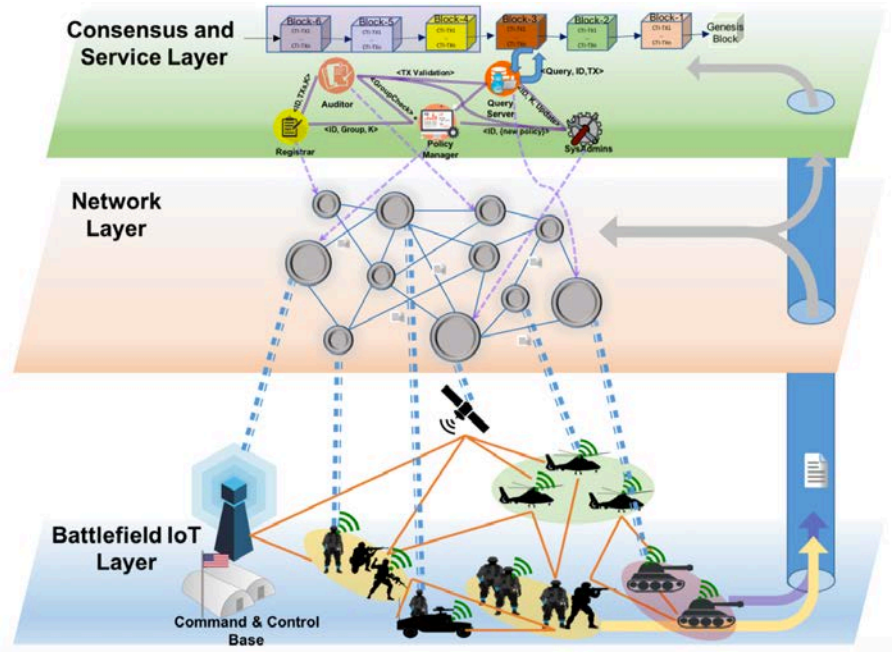


Figure 28: Blockchain-enabled IoBT Architecture

a. Battlefield Sensing Layer

Given the technological revolution happening, the characteristics of modern warfare are expected to similarly evolve and the warfighters would be looking for higher cyber capabilities in the battlefield to stay ahead of adversaries. So, use of tactical computing resources, networking-capable sensors, unmanned systems, and wearable devices are of high demand to better understand such a complex and competitive environment, while facilitating the actors to make better and faster decisions. With the help of collaborative sensing of the environment and short/long-range communication technologies, battlefield IoT nodes should be exchanging mission-specific information with each other. Hence, this battlefield-sensing layer is constituted of all the military objects that collaboratively operate in the physical warfare space for sensing and monitoring adversarial activities. Not only environmental sensing, but also monitoring troop progress, inventory management, and health checkup of individual soldiers is critical to the success of mission. With this kind of information, necessary assistance can be arranged at the right time and potentially reduce the chances of mission failure. Consumer devices such as smart phones, health trackers, autonomous vehicles, and so on come with their own sensing modules and when they are deployed on a large scale in the battlefield, it can provide military Intelligence, Surveillance, and Reconnaissance (ISR) capabilities through crowdsensing techniques [115].

The commercial market for IoT has been introducing plenty of powerful and network-capable devices built on top of traditional processing hardware (microprocessor and microcontroller). Additionally, several innovative hardware solutions and multi-core CPU-based computing architectures are emerging to meet the computationally heavy services and energy efficiency requirements. Military equipment and soldiers are subjected to carrying a variety of sensors and actuators that can measure attributes related to locations, speed, health, brightness, temperature, pressure, electro-chemical and electro-mechanical properties, etc. All the network-capable

entities in the battlefield may not follow the same standard of communication. Different mediums and protocols like WiFi, Infrared, Bluetooth, Near-Field Communication (NFC), X10, cellular networks, and ZigBee. etc., are used to disseminate the sensed data among each other. Thus, interoperability and protocol standardization are crucial challenges in this layer to achieve operational efficiency in the IoBT network.

b. Network Layer

The sole purpose of the Network layer is to transmit and capture the network-centric transactions occurring at the Battlefield-Sensing layer. The common-use cases of IoT, like smart home and smart city, operate off infrastructure-based connectivity, where the nodes are connected to each other via a gateway or access point. However, this will be not feasible in the case of the battlefield because connectivity to cellular networks or any base stations may not be available throughout the period of operation. Therefore, D2D communication [129] is of top choice for exchanging information with each other. In such a scenario, the networks created in the battlefield are weakly connected and volatile in nature, which means the topology may not sustain longer use. Also, at times the network may get segmented if the military troops need to be separated from each other to subdivide the mission tasks. Furthermore, the dynamic network created can be categorized to constrained and unconstrained networks. Devices in the constrained network may have power, memory, and data rate limitations. In addition, they may operate in an unlicensed spectrum where interference is high. Thus, the network entities need to optimally tune their physical transmission-related parameters to maintain the network connectivity.

Irrespective of the connectivity challenges, the nodes in the network are intended to collect valid transactions and propagate them toward the closest nodes that participate in Blockchain consensus. It is not expected that all nodes of the network will be serving as a full node, which typically stores the complete state of the Blockchain. Rather, a subset of the nodes can be selected to be full nodes and the rest of them can act as endorsers, who verify the validity of transactions and endorse them so that the full nodes can trust those transactions. An overlay network among full nodes can be designed to maintain strong connectivity among them so that the network layer performance of the framework can be improved. Designing such an overlay network is feasible in a static network [130]; however, it is challenging in the case of highly

Figure 29: IoT Network Standards and their Trade-offs

dynamic networks like in the battlefield environment.

The architecture shown in Figure 28 depicts that some military entities (from Battlefield-Sensing Layer) formulate the core P2P network among each other for maintaining the shared ledger. In the figure, the nodes are represented as varying sized circles, where the sizes represent the heterogeneity and resourcefulness of the participating nodes. This means that larger circles represent a node in the military network that has more resources (computational, storage, networking capability, energy, etc.). To establish communication between each node, the nodes are capable of using the existing wired/wireless standards that include: ethernet, WiFi, Bluetooth (Low Energy), NFC, ZigBee, EnOcean, WiMax, LTE, Cellular, and LoRaWAN, etc. Each of these standards have their own pros and cons based on their maximum data rate, power consumption, and coverage radius [131] as shown in the Figure 29. The communication standards of IoT space are not mature yet in terms of addressing the connectivity issues.

c. Consensus and Service Layer

The role of the Consensus and Service layer is to employ common agreement mechanisms for accepting the valid set of network-centric operations in the form of transactions that happened in the Battlefield Sensing layer. At the same time, maintaining the total or partial ordering of the transactions is important for the future audit purpose. By ingesting the messages in the Network layer, the nodes in this layer filter as well as check the validity of each transaction, and eventually inserts them into the system ledger when the majority consensus on the authenticity of the transactions is achieved. There are a number of consensus protocols proposed from the distributed system's point-of-view. Consensus in distributed systems is equivalent to the Byzantine General's Problem, which has been studied extensively in the past literature [132, 133]. Such protocols have helped immensely to design fault-tolerant distributed systems such as distributed storage, P2P sharing, multi-party computation and also cryptocurrency technology. The communication overhead and tolerance to Byzantine faults in Paxos [134] were its main demerits, which are addressed by Castro et.al [135] in the work on the practical Byzantine fault-tolerant (PBFT) model. However, PBFT is not completely decentralized like Paxos because it requires a round-wise coordinator to impose ordering of transactions and its stability depends on the fact that it can tolerate $f < n/3$ Byzantine nodes. Also, the communication delay in the PBFT is bounded, which makes it a weakly synchronous protocol.

The Proof of Work (PoW) consensus protocol has been adopted in several cryptocurrency Blockchains that operates using the computing power of the participating nodes. However, it introduces a *tradeoff* among consensus convergence, transaction throughput, and security. In addition, the PoW consensus consumes more energy for repeatedly computing the cryptographic hash function to mine the blocks. To avoid such constraints, Proof of Stake (PoS) [135,136] consensus uses the ownership of resources to asynchronously select nodes, who will create blocks. Hence, a node with a higher amount of assets reserved for consensus will have a higher chance on an average to create blocks. Alternative consensus mechanisms that are also considered for the Blockchain, such as Intel's Proof of Elapsed Time (PoET), and Byzantine fault tolerant variants. However, none of these consensus models were designed by considering the distributed and intermittent nature of a battlefield network. Also, the nodes need to serve for certain important roles for maintaining the common distributed ledger and network of consensus

participants irrespective of the dynamism in the military network. The necessary roles considered here are described briefly in the following:

1) Registrar: This particular role is intended to serve for registration and key distribution purposes, when any newly introduced military component or personnel appears in the battlefield environment. The registered entities can seamlessly check the identity details of each other to verify and sign the transactions/blocks. The registrar can keep track of the access policy changes by querying the policy manager.

2) Policy Manager: In the battlefield environment, the entities have different rankings and so they might have higher privileges compared to low ranked entities. Hence, it is re- quired to define and enforce such kinds of policies in the military networks. The role of the policy manager is to provide an interface to define policies related to Blockchain transactions, consensus, network connectivity, and privacy aspects. In addition, the block/transaction validation rules can also be managed through the policy manager, which helps to fine tune the Blockchain performance.

3) Auditor: This role is created to interrogate and keep an eye on the transactions. The auditor in the framework collects the network-centric transactions occurring in the military network and audits them on the fly by using the Blockchain receipts. To maintain the distributed environment, there can be more than one auditor in the network, which can be achieved by executing a smart contract that permits a set of tasks to execute as an auditor. The auditor is also responsible for keeping track of valid transaction blocks and corresponding mission-specific transactions by tagging with block metadata. The accepted blocks need to be pre-processed by extracting transactions and metadata information for ordering purpose.

4) Query Server: One or more participants take the role of query server to undertake requests from military entities, who can validate themselves to be authentic member of the troop. Hence, the purpose of query server is to validate the requesting party's identity before responding with necessary information and also to ensure the responses do not possess any private information. By distributing the query manager roles to multiple consensus nodes may allow tackling the scalability issue effectively.

5) System Administrator: The SysAdmin usually acts like a membership manager, which is responsible for evaluating the necessary requirements of the nodes, who wish to participate in the Blockchain consensus and take charge of one or more roles as defined here. It also defines contracts to track participation duration, reliability, and incentives components to engage the nodes to voluntarily dedicate their resources for offering secure Blockchain services in the battlefield.

3.5.5 Applications of the Blockchain-based IoBT Architecture

Although blockchain is providing a distributed ledger system that has many needed security features and functionalities, this could help the much needed security for cyber operations in military battlefield, such as, auditability of historical information, assurance of data provenance,

guaranteed variability of integrity violations of historical data, and information tampering detection. Furthermore, blockchain has both cost effectiveness merits as well as transparency features, making it an appealing system for military cyber operations as described in the following cases.

Generation of cyber assets - Blockchain can be used to generate cyber assets that will enable applications which rely on direct interaction between customers and assets. The Blockchain system can aid in ensuring the issuance processes, transaction processing, and housing of cyber assets/identities.

Transfer of ownership of cyber assets - A Blockchain system allows transfer of cyber assets between owners by leveraging the immutability property of Blockchain so that once a transaction is committed, it cannot be reversed. Any changes will have to be appended and will not alter an already validated transaction, thereby ensuring non-reversibility of transfer of ownership.

Transparent and assured data provenance - Every operation on the cyber asset is encoded in the Blockchain trans- action using a publicly available and immutable ledger. The Blockchain system ensures that provenance of every operation on the cyber asset is recorded and traceable.

Verifiability and audit - The distributed ledger keeps track of transactions pertaining to creation and transfer of cyber assets. The tamper-resistant property of the ledger facilitates variability and audit of operations.

Military cyber operations - Ensuring traceable and tamper-evident accountability and auditability of C2, logistics, and other critical mission data among international partners is paramount as our future operations involve the convergence of multiple domains and heavily contested cyberspace. Centralized or homogenous information systems and databases must evolve distributed, disintermediated, and secure capabilities. As such, trust with respect to operations involving international entities must not be rooted in one single entity. Trust must be decentralized and built around robust, innovative cryptographic paradigms transcending the traditional Public Key Infrastructure (PKI) typically used in most homogenous enterprises.

An innovative, distributed trust and identity management mechanism is a crucial for enabling assured identification, authentication, and authorization in such a way that would further allow disintermediated accountability and auditability. Emerging Blockchain and distributed ledger technologies as a whole demonstrates the potential of a truly distributed and disintermediated mechanism for achieving above needs. The current production application of cryptocurrencies using public Blockchain has already shown the potential of decentralization to allow customers to perform monetary transactions seamlessly and maintain the ledger at the same time. The nuances of disintermediated international partnerships and information exchange involve some mutually exclusive research and development challenges distinct from the permissionless and public implementations of Blockchain.

3.5.6 Performance Measurement and Evaluation

To evaluate the performance of permissioned blockchain in the battlefield sector we present a proof of concept that implements the Hyperledger Sawtooth platform [137]. The Sawtooth consists of three major components: a validator running a consensus protocol called Proof of Elapsed Time (POET) that is responsible for creating a block of approved transactions, a REST API which handles communications between clients and data transference through HTTP/JSON, and a transaction processor which validates and verifies transactions. We select Sawtooth because it provides a role-based identity manager to enforce data access policies suitable for our battlefield scenario. With Sawtooth chosen, we aim to test how well this variant can perform in an IoBT scenario. Military battlefields can be setup with under different constraints and resources, making a flexible blockchain platform critical. Success can be measured by how many transactions are performed in the blockchain as well as how many of those transactions are verified. We conduct our simulations on the Common Open Research Emulator (CORE), a software used to simulate various types of networks [138]. Evaluations are based on a series of tests that measure how well Sawtooth would perform under different parameter changes. All tests include one genesis node and five client nodes. Each simulation presents a different combination of parameters, changing bandwidth and transaction rate.

Experimental Setup:

Due to the importance of a network setup in military environments, we test using three topologies: fully connected, mesh, and tree. For reference, each topology can be seen in Figure 30. Each setup can be directly related to a particular military scenario, from command center setups to battlefield ad-hoc networks.

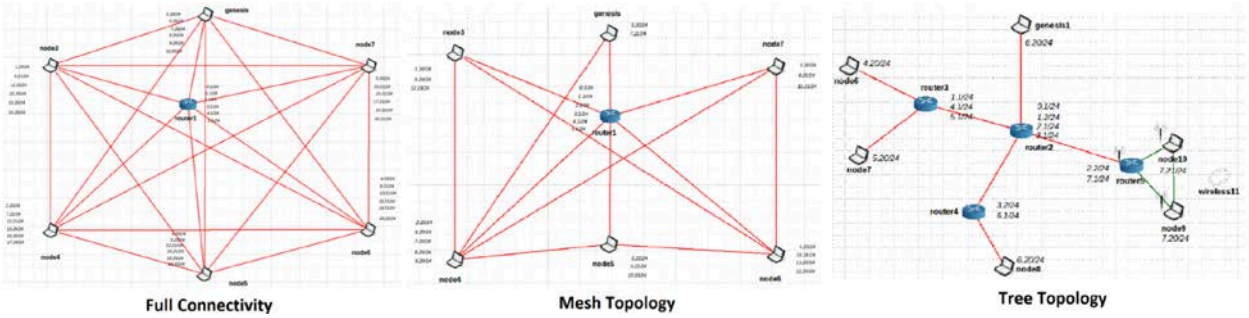


Figure 30: IoBT Relevant Topologies for Evaluation

Take the following scenario: a six node battlefield is setup with the intent of successful data reconnaissance under limited resources. Client nodes, which excludes the genesis, actively send image information to the blockchain for processing. To test this: metadata from an image was extracted using a Python script, then bundled together as a single transaction for a validator node, stored until a specified number of validated transactions was reached, then added to the ledger. Each simulation occurs for 1 hour, under three different bandwidth options: 1 Gbps, 100 Mbps, and 64 Kbps. A traffic capture was taken for each test, where metric data was separated into two categories: network maintenance packets and image data packets.

For every bandwidth option and topology pairing, a total of 10 tests were conducted. Each test changed the rate at which transactions were sent to a validator node in the network. These rates included 1, 10, 25, 50 and 100 transactions at every 2 and 5 seconds. This yielded a total of 90 tests for analysis.

Ideal scenario of unlimited link bandwidth:

CORE's term for an unrestricted bandwidth is known as unlimited. However, the performance of this term is completely dependent on machine hardware. To factor out this dependency, we compare the unlimited bandwidth to that of 1 Gbps, the highest numerical value available in CORE. Table I shows the number of transactions sent in a 10 minute period with a rate of 50 transactions at every 2 seconds for all topologies under each bandwidth option.

As seen in the table, there is some deviation between the unlimited option and 1 Gbps. For every topology, the 1 Gbps was able to successfully deliver 20 to 30% more data packets, and thus more transactions. This can be attributed to the routing and communication protocols for unlimited and 1 Gbps. Where the 1 Gbps bandwidth has built in packets delays and optimized routing, the unlimited does not. For this reason, the testing set we will use in this paper includes only the three bandwidths specified above.

Table 5. Data packet comparison for "unlimited" and 1 Gbps BW

	Full	Mesh	Tree
Unlimited	93169	77555	55689
1 Gbps	114477	103679	84012

Network Maintenance

Packets captures for each simulation are split into two categories: maintenance and data. Network maintenance refers to those packets which help maintain constant communication and operability for the blockchain. Packet types include ARP, MDNS, and TCP acknowledgments. Using figure 31, data shows that for any topology and transaction rate, under the 1 Gbps and 100 Mbps bandwidth allotment, maintenance packets constitute an approximate 49-51% of the total capture. However, there is a performance difference between those two topologies, dependent on bandwidth. With 1 Gbps, the mesh topology in combination with 2 second transactions consistently used the highest number of packets. With 100 Mbps, the full topology in combination with 2 second transactions managed to use more packets.

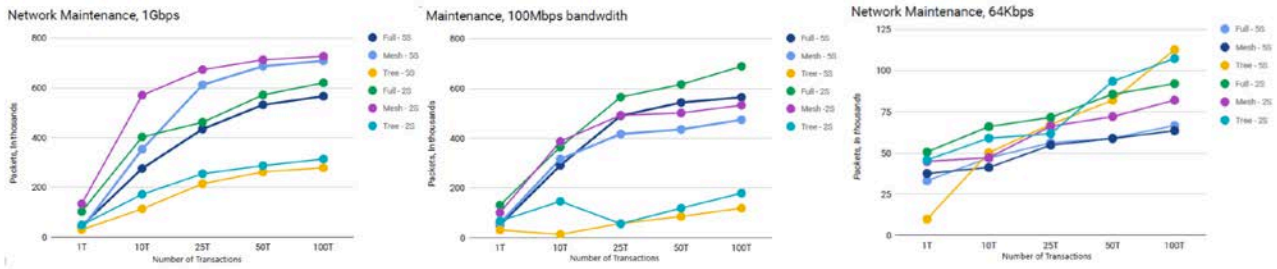


Figure 31: Maintenance packets for 1Gbps, 100Mbps, and 64Kbps simulations.

There are two main reasons for this. In the case of the mesh topology, the combination of bandwidth and node placement made for an efficient system, where a high level of transactions were sent and validated. When observing the tree topology, the number of packets rose due to a buffer constraint contained in Sawtooth's validation scheme. When too many transactions are

sent to the buffer and the max value is reached, transactions are lost and thus, the network must resend packets.

The major outliers for these simulations are those that use the 64 Kbps bandwidth. The first point, is that the amount of transactions sent is significantly lower, which leads to an assumption that network maintenance packets would also decrease. However, the maintenance percentages rose to an average range of 54%-70%. This is due to the bottleneck created by the low bandwidth. Transactions are lost as the buffer cannot process in time, and thus the network becomes congested with retransmission of acknowledgments, Sawtooth's native heartbeat function, and others. Figure 31 demonstrates the amount of packets needed to maintain network functionality under a 64 Kbps bandwidth setup.

Total Data Communication

In this section, we look for packets that relate specifically to image metadata transmission or propagation of the blocks in the chain.

1) *Topology:* It is evident that node placement and connectivity in these simulations play a factor. Overall, the full and mesh topologies functioned at very similar rates and performance peaks. Having 1 Gbps bandwidth allows the mesh topology to perform better than the full topology, as seen in Figure 31. This is followed by the tree topology, the weakest performer.

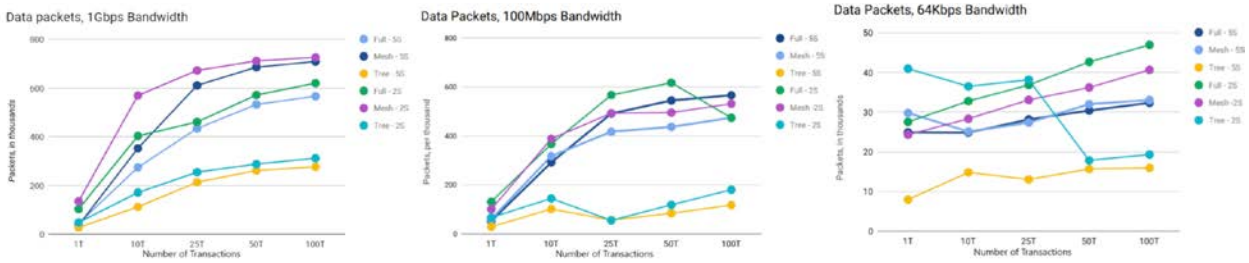


Figure 32: Number of total data packets sent in different type of links and topology

When 100 Mbps bandwidth was available, the mesh and full topologies had overlapping results. Figure 31 shows how for all simulations that send "x" amount of transactions every 2 seconds, mesh performs better at certain rates like 10 transactions and 100 transactions, versus the full topology at all other points. The simulations that send "x" amount of transactions every 5 seconds show the mesh topology as the most effective choice, with a full topology trailing closely behind. For all tests under the 1 Gbps and 100 Mbps bandwidth, the tree topology consistently had the lowest rates of data throughput. This however, is not true for the 64 Kbps captures. Referring to Figure 31, the tree topology had the highest starting point for simulations that sent transactions every 2 seconds and in a more common result, the lowest starting point with simulations that sent transactions every 5 seconds. However, as the rate increased, the throughput decreased significantly.

2) Bandwidth: There are three major points of focus that the data reflects in Figure 31. First, is that the highest amount of data throughput is observed under the 1 Gbps bandwidth; the highest value being around 700K data packets at 100 transactions every 2 seconds. Next, is that for 1 Gbps and 100 Mbps bandwidths, the throughput rose steadily until the 100 transaction marker. At that point, the tree topology under 100 Mbps had a sharp decline. This coincides perfectly with the network maintenance seen in the previous subsection, as more network failures caused

this topology to underperform. Finally, the 64 Kbps captures are highly variable. In some instances, 1 transaction every 5 seconds was able to out- perform 10 transactions every 2 seconds, as noted by mesh and tree in figure 31 for 64 Kbps. We can attribute this to a combination of both node instability and network processing failures due to increasing congestion of transaction validation. Additionally, the number of transactions is significantly smaller when compared to the two previous bandwidths.

Total Validated transactions - Throughput

After detailing how the topologies and different bandwidths interact, we must also denote the success rate (i.e., the percentage of successfully validated transactions). This section uses 10 minute tests for all topologies under two bandwidth parameters with an increasing transaction rate that ranges from 1 to 10,000. To find the success rate of each combination, we take the number of successfully validated transactions and divide by the total transactions sent between all nodes. We will denote it as a percentage.

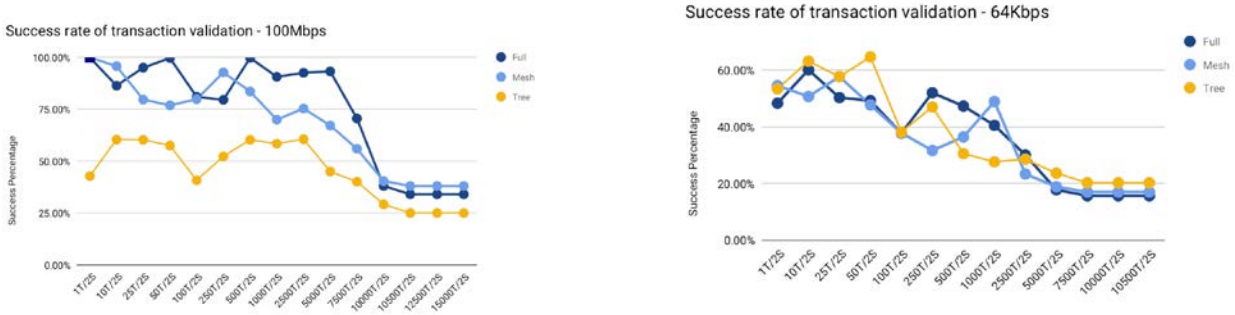


Figure 33: Actual throughput (a) 100 Mbps link (b) 64 Kbps link

1) 100 Mbps Bandwidth: In Figure 32(a), for the full topology, we start with a 100% success rate, then a small decline as the transaction rate increases. This increase and decrease of success is consistent, up until 1000 transactions. After this, we observe a stagnant success percentage, followed by a sharp decline. At 7,500 transactions or more, this topology seizes being effective with a final success percentage of 38%.

The mesh topology shows a very different rate of decline when compared to the full topology. We start at 100%, then begin a steady decline as the transactions increase. At 250 transactions, a small increase in success rate is seen, making it the highest value before a steady decline of success as transaction rates increase. We end the simulations at a success rate of 40%, just above the full topology.

As demonstrated by the network maintenance chart in the previous sub-section, the tree topology success rate is much lower than the rest. The highest success percentage observed was 80% with transactions in the range of 10 and 250, excluding 100 transactions. This topology always under-performs when we compare it to the full and mesh, ending the simulations at a 25% success rate.

2) 64 Kbps Bandwidth: Figure 32(b) highlights that when the bandwidth is decreased to 64 Kbps, the success rate never climbs above 65%. The rate of decline is much more steady in these tests with a very surprising result: the tree topology is able to have the highest performance marker at the end of the set of tests. Although the tree topology is highly variable, it is able to end at a higher success rate due to the combination of network routing, packet delays, and topology setup. Because much of the routing redundancy is reduced in the tree topology and the network

packet bottleneck is reduced, the Sawtooth validator queue is able to take more incoming transactions rather than rejecting them.

For the full topology, the highest success rate is approximately 54% at the lowest transaction rate possible and 64 Kbps bandwidth. This success rate is observed approximately between 5000 and 7500 transactions when 100 Mbps bandwidth is available. The mesh topology is very much similar to full, showing an approximate 54% success around 7500 transactions with 100 Mbps of bandwidth. Between full and mesh, the data shows that the full topology had the highest number of success data points overall.

The tree topology demonstrates an interesting set of results. Although volatile still, it managed to end the set of tests with the highest success marker. However, the performance of this topology is still lower when compared to the full setup. Because it does better than all other topologies before 100 transactions at 64Kb bandwidth, it is a viable option for a very small subset. An additional point is that the tree success rate remained somewhat consistent between both bandwidth values. This mean that if more bandwidth is available, the tree topology will run at a capped performance peak, with the decay of success extending further out as the bandwidth grows.

It is worth nothing that neither of the charts show tests reaching a complete lack of transaction processing. The main cause of non-validation is due to a queue maintained by all validators. When the queue is full, the transaction is dropped and therefore lost. The logic then follows that while the queues may always be full, there exists a small percentage that is validated, as noted by the high transaction rates in both the 100 Mbps and 64 Kbps charts.

Remarks

With the data available, we can conclude several things. First, is that a decentralized network approach (full and mesh) are the most effective when it comes to a blockchain setup. We are able to sustain a more efficient system on these platforms. When it comes to the most ideal setup, we consider two things: data throughput and success of validation. Figure 30 showed that the mesh topology was able to produce the greatest amount of data packets over a 1 hour time period, however, the success rate of the mesh topology was lower overall than that of the full topology for both 100 Mbps and 64 Kbps bandwidth. The ideal setup then becomes a full topology with the highest success rate demonstrated at 1 and 5,000 transaction(s) every 2 seconds when 100 Mbps bandwidth is available or a full topology between 1 and 250 transactions every 2 seconds when 64 Kbps bandwidth is present.

Secondly, we are able to conclude that the Hyperledger Sawtooth, although configurable, is operational when higher resources are available. While we are able to perform tests under small time constraints, the amount of storage and band- width needed over time is too significant when we consider the low-resource devices that a soldier may carry. Furthermore, these tests approximate small military operations. When scaled to a bigger size, there is risk that the blockchain platform may not provide a 100% operational guarantee to maintain the ledger due to resource constraints.

Other Factors: There are other testing parameters which can play a significant role in the metric data produced. In all of these simulations, the nodes are stationary and not moving. Nodes that

are able to travel in and out of range of a wireless network might significantly alter the data, as Sawtooth may behave abnormally or respond in a way such that the network would collapse. We are not changing the number of nodes across the different simulations, which may also present an interesting side effect to the data.

Additionally, the POET consensus method is one of two different methods that may be utilized with Sawtooth. In this configuration, we do not observe any form of stale blocks, as the rules specify a block must be formed once the minimum number of validated transactions is achieved. This cleans up network propagation where as other consensus methods may not be as efficient.

3.5.7 Concluding Remarks

The ideal and secure communication system for military battlefields requires the synchronization of resources and protocols. With attackers being always one step ahead, innovative mechanisms must be devised that allows soldiers to remain effective on the battlefield. We understand that the traditional centralized networks in battlefields can be costly to maintain and have security demerits, so a blockchain integrated platform for IoBT can be of potential choice. Therefore, we proposed a 3-layer blockchain empowered IoBT architecture that can help securing military cyber operations and mission related data transactions in a tactical environment. Since there are not enough studies to determine the performance of current state-of-the-art blockchains in IoBT context, we integrated a permissioned blockchain with a distributed IoBT environment to achieve necessary security needs and evaluated the network constraints on the blockchain under various topologies. This deployed system is decentralized in nature and can meet the security requirements of military operations. The simulations show that a particular permissioned variant of blockchain is has potential to secure the battlefield network given we allocate more computing resources in the battlefield. It also shows promise for other similar blockchain implementations to be applicable. In the future, variety of blockchain platforms can be integrated in the IoBT testbed to analyze and test the effectiveness as well as readiness of each variant. In addition, the mechanism to tune the inherent attributes of each variant to meet the specific operational needs of the tactical environment

4 RESULTS AND DISCUSSIONS

Although the benefits that a Blockchain-enabled IoBT system can offer to plan and execute military missions are numerous and compelling, several challenges exist and need to be addressed for successfully realizing the framework. In the following subsections, we categorically discuss the challenges involved while implementing each layer of the proposed architecture.

A. Battlefield Sensing Layer Challenges

When a higher number of sensing devices are collectively working in the battlefield, the amount of Blockchain transactions generated will also be increasingly high. It raises several issues and bottlenecks for the overall system, which we brief in the following.

1) Fine/Coarse Grained Sensing: In practice, the physical entities in the battlefield arena may carry more than sensors/actuators, which have different sensing purposes and more than one node may have communication capability even if they are tied with the same physical object. In this situation, the number of data streams to transmit the information increases and thus the transaction count for a Blockchain network layer grows accordingly. Since adding a block of transactions into the distributed ledger requires consensus to be achieved, which is a time-consuming process, it can introduce performance overheads too. Therefore, fine-grained sensing might improve the accuracy, but the Blockchain infrastructure may not be able to handle such a large amount of transactions. Hence, multiple sensing streams may need to be aggregated and a balance between fine, and coarse-grained sensing needs to achieved, which can help in attaining optimal Blockchain performance.

2) Interoperability: Military equipment and hardware used in the battlefield are typically manufactured from different vendors and the interoperability between these IoT devices is overlooked, which plays a major role in producing standardized transactions for the Blockchain platform. Furthermore, sometimes the high-end military equipment gets some critical parts replaced upon any damage, thus it may change the identity information of the device and require following a different standard to operate. Since it is difficult to have all the equipment follow the same communication standard, the important challenge here is to design a secure interoperable layer that can be generically used by every network-capable node to create globally-acceptable Blockchain transactions.

3) Energy Efficiency: Power has a critical role in engaging military equipment to operate well in harsh situations. Sensing the terrain and physical conditions consumes energy. And, sending the data to other peers as well as the Blockchain network costs additional power. So, a right balance on sensing and transmission tasks is required as per the mission needs, and transaction frequency should be optimally selected for the Blockchain framework to minimize the overall energy cost.

B. Network Layer Challenges

The network layer creates the backbone of our Blockchain-enabled IoBT architecture, where the transaction gathering, block propagation, and Blockchain service-related communications take place. Hence, maintaining strong and reliable connectivity is important to avail all the benefits of Blockchain. The challenges involved in this layer are briefed below.

1) Participants Selection: To have a distributed IoBT platform, it is important to have a consistent and reliable set of nodes, who can serve to maintain the Blockchain. Traditional IoT environment has ubiquitous network connectivity with which the nodes can send their data to a centralized cloud server that will perform the necessary analytics tasks and monitor the environment accordingly. However, the IoBT devices operate on stringent conditions, such as low computational capability, lack of network connectivity, low/no energy supply, and so on. Thus, it is necessary to have a robust selection criteria to select nodes that can keep the Blockchain network active until a fixed period of time is reached. Also, mechanisms need to be established to periodically offload the responsibilities of existing nodes to another selected set of nodes for keeping the network alive until the mission completes.

2) Dynamic Network Topology: Military missions are usually very dynamic in nature, where battlefield equipment and soldiers always change their locations. Thus, the network created by them will have both spatial and temporal topology variation. Furthermore, the energy drainage

on the devices may even segment the network to multiple pieces. Considering these challenges, the adaptable communication protocols for the distributed ledger technology should be established to maintain consistency and ordering of transactions in Blockchain.

3) Blockchain Parameter Tuning: The network layer mostly handles the sending and receiving of transactions and blocks among each node. Given the established tactical network may have a limited bandwidth, the size of transactions and blocks need to be optimally chosen so that overall latency in the consensus process can be minimized.

C. Consensus and Service Layer Challenges

Establishing the network of military things is not sufficient unless a robust distributed consensus mechanism is in place to maintain the Blockchain state. The traditional consensus algorithms have limitations to adapt in the IoBT environment. Therefore, applicability of different Blockchain consensus models needs to be investigated and revised appropriately to serve the needs of a Blockchain-enabled IoBT framework.

1) Choice of Consensus Mechanism: At present, there are a number of consensus schemes proposed for both public and permissioned Blockchain platforms. Among those, PoW consensus adopted in the public Blockchain of Bitcoin and Ethereum has scalability constraints that supports ~3 and ~12 transaction per second respectively. However, adoption of public Blockchain in battlefield zones may not be possible due to lack of connectivity to the Internet. The Permissioned Blockchains like Hyperledger opt for PBFT consensus that can achieve significantly higher throughput at low latency. In addition to BFT variants, a number of federated consensus models such as Stellar [39], Ripple, and Algorand [140] are proposed. Thus, it is challenging to find the right consensus model that will best work for the IoBT Blockchain, while considering the energy constraints, sparse connectivity issues, and transaction throughput requirements. Also, the dynamism of battlefield nodes poses the question of "who will be responsible to hold the Blockchain data (authenticated transaction data) permanently to make it available throughout?"

2) Number of Nodes: As the nodes in battlefield enter and exit the network sporadically, the consensus requires a stable number of nodes to confirm the Blockchain state. If an insufficient number of nodes operate at the consensus layer, consistency of the Blockchain can be compromised. Although, fewer consensus participants can improve the transaction throughput, it may not be secure enough to prevent malicious exploitation of the consensus process.

3) Performance and Privacy Considerations: Traditionally, each node verifies every single transaction in the Blockchain framework in parallel before the block mining occurs. This becomes a bottleneck when it comes to improving the scalability and transaction throughput in any IoT system. In addition to devising lightweight consensus mechanisms, several other techniques like sharding, off-chain computation, and state channels [141] are considered for improving the Blockchain performance. However, it will be important to investigate their usefulness in improving the battlefield-specific Blockchain. In addition, privacy of transactions is critical to maintain when the ledger is shared among the participating military nodes.

5 CONCLUSIONS

In this report, the key contributions are a) Blockchain simulator for IoBT environment, b) Lighweight Blockchain with sharding, c) Techniques for optimizing memory pools against flooding attacks and framework for characterizing blockchain based systems for accurate systems.

We present a Blockchain simulator for testing and evaluating consensus algorithms in a realistic and configurable network layer. This paper introduces and shows a generalized consensus protocol method that can be used to model various consensus protocols. By observing the ability for the system to operate in various configurable network topology's we begin to study under what conditions the system fails to operate. This can occur when portions of the network do not allow the peer nodes to quickly message in participation of transactions. The unique characteristics of the simulator are a general consensus algorithm operating in a realistic and configurable network environment. The discrete event simulation engine allows to specify the consensus algorithm operations at faster than real-time fidelity without loss of scalability. In future work, we plan to implement and calibrate additional consensus algorithms to accommodate bootstrapping scenarios and evaluate performance in diverse network configurations. Future work with this model and simulation will include the effects of the consensus special state using the consenPauseState variable, which will be modeled to get a better comparative test between different consensus protocols.

We proposed a 3-layer blockchain empowered IoBT architecture that can help securing military cyber operations and mission related data transactions in a tactical environment. Since there are not enough studies to determine the performance of current state-of-the-art blockchains in IoBT context, we integrated a permissioned blockchain with a distributed IoBT environment to achieve necessary security needs and evaluated the network constraints on the blockchain under various topologies. This deployed system is decentralized in nature and can meet the security requirements of military operations. The simulations show that a particular permissioned variant of blockchain is has potential to secure the battlefield network given we allocate more computing resources in the battlefield. It also shows promise for other similar blockchain implementations to be applicable. In the future, variety of blockchain platforms can be integrated in the IoBT testbed to analyze and test the effectiveness as well as readiness of each variant. In addition, the mechanism to tune the inherent attributes of each variant to meet the specific operational needs of the tactical environment

Finally, we developed and presented a simulator, called FastChain, built in NS-3 which simulates the battlefield scenarios with military applications which connects tankers, soldiers, and drones to form IoBT. The simulator uses the sharding enabled blockchain for trustworthy IoBT operations. Resource constraint IoBT devices form a group to participate in sharding enabled blockchain for IoBT scenarios. Researchers, educators and policymakers working on IoBT or similar scenarios can use the FastChain simulator and evaluate their systems.

6 REFERENCES

[1] Croman, K., C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gün Sirer,D. Song, and R. Wattenhofer. 2016. "On Scaling Decentralized Blockchains". InFinancial Cryptog-raphy and Data Security, edited by J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, andK. Rohloff, pp. 106–125. Berlin, Heidelberg, Springer Berlin Heidelberg.

[2] Segent, N. 1997. "Performance evaluation of a consensus algorithm with Petri nets". InProceedings of theSeventh International Workshop on Petri Nets and Performance Models, pp. 143–152. IEEE.

[3] Gervais, A., G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. 2016. "On the security andperformance of proof of work blockchains". InProceedings of the 2016 ACM SIGSAC conference oncomputer and communications security, pp. 3–16. ACM.

[4] Docker 2018, Aug. "Docker Documentation". https://docs.docker.com/.

[5] Ongaro, Diego and Ousterhout, John 2013. "In search of an understandable consensus algorithm (extended version)".

[6] Shelat and C.H. Shen, "Fast two-party secure computation with minimal assumptions." ACM CCS 2013, pp. 523-534, 2013

[7] V. Varghese, S. S. Desai, and M. J. Nene, "Decision Making in the Battlefield-of-Things," *Wireless Personal Communications*, pp. 1–16, 2019.

[8] A. Adebayo, D. B. Rawat, L. Njilla, and C. A. Kamhoua, "Blockchain-enabled information sharing framework for cybersecurity," *Blockchain for Distributed Systems Security*, p. 143, 2019.

[9] D. B. Rawat, V. Chaudhary, and R. Doku, "Blockchain: Emerging Applications and Use Cases," *arXiv preprint arXiv:1904.12247*, 2019.

[10] D. B. Rawat and K. Z. Ghafoor, *Smart Cities Cybersecurity and Privacy*. Elsevier, 2018.

[11] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," http://bitcoin. org/bitcoin. pdf," 2008.

[12] D. B. Rawat, L. Njilla, K. Kwiat, and C. Kamhoua, "iShare: Blockchain-based privacy-aware multi-agent information sharing games for cybersecurity," in *2018 International Conference on Computing, Networking and Communications (ICNC)*, pp. 425–431, 2018.

[13] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 931–948, 2018.

[14] T. Rocket, "Snowflake to Avalanche: A novel metastable consensus protocol family for cryptocurrencies," 2018.

[15] https://ipfs.io/ipfs/QmUy4jh5mGNZvLkjies1RWM4YuvJh5o2FYopNPVYwrRVGV.

[16] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools*

for network simulation, pp. 15–34, Springer, 2010.

[17] R. Doku, D. B. Rawat, M. Garuba, and L. Njilla, "LightChain: On the Lightweight Blockchain for Internet-of-Things," in *Proceedings of 2019 IEEE International Conference on Smart Computing (SMARTCOMP-2019)*, pp. 444–448, 2019.

[18] Danda B. Rawat, Principle Investigator of FastChain Project and Simulator: URL: https://sites.google.com/site/fastchain4shardingiobt.

[19] C. Wee, "Audit logs: to keep or not to keep?" in *Recent Advances in Intrusion Detection*, 1999. [Online]. Available: http://www.raid-symposium.org/ raid99/PAPERS/Wee.pdf

[20] A. Ahmad, M. Saad, M. Bassiouni, and A. Mohaisen, "Towards blockchain-driven, secure and transparent audit logs," in *International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous, New York City, NY, USA*, 2018, pp. 443–448. [Online]. Available: https://doi.org/10.1145/3286978.3286985

[21] C. Ringelstein and S. Staab, "DIALOG: distributed auditing logs," in *IEEE International Conference on Web Services, ICWS, Los Angeles, USA*, July 2009, pp. 429–436. [Online]. Available: https://doi.org/10.1109/ICWS.2009.50

[22] A. Ahmad, M. Saad, M. Bassiouni, and A. Mohaisen, "Towards blockchain-driven, secure and transparent audit logs," in *International Workshop on Distributed Ledger of Things (DLoT)*, Nov 2018.

[23] M. Saad and A. Mohaisen, "Towards characterizing blockchain-based cryptocurrencies for highly-accurate predictions," in *IEEE Conference on Computer Communications Workshops, INFOCOM Workshops, Honolulu, HI, USA*, April 2018, pp. 704–709. [Online]. Available: https://doi.org/10.1109/INFCOMW.2018.8406859

[24] M. Saad, L. Njilla, C. A. Kamhoua, and A. Mohaisen, "Countering selfish mining in blockchains," *CoRR*, vol. abs/1811.09943, 2018. [Online].
Available: http://arxiv.org/abs/1811.09943

[25] G. Nguyen and K. Kim, "A survey about consensus algorithms used in blockchain," *JIPS*, vol. 14, no. 1, pp. 101–128, 2018. [Online]. Available: https://doi.org/10.3745/JIPS.01.0024

[26] ClearVillage, "Clearvillage," 2018. [Online]. Available: http://www.clearvillageinc.com/

[27] J. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, ser. Lecture Notes in Computer Science. Springer, 2001, vol. 2050. [Online]. Available: https://doi.org/10.1007/3-540-45318-0

[28] M. Saad, M. T. Thai, and A. Mohaisen, "POSTER: deterring ddos attacks on blockchain-based cryptocurrencies through mempool optimization," in *Proceedings of Asia Conference on Computer and Communications Security, ASIACCS, Incheon, Republic of Korea*, June 2018, pp. 809–811. [Online].
Available: https://goo.gl/4kgiCM

[29] P. Berenbrink, T. Friedetzky, P. Kling, F. Mallmann-Trenn, L. Nagel, and C. Wastell, "Self-stabilizing balls & bins in batches," *CoRR*, vol. abs/1603.02188, 2016. [Online]. Available: http://arxiv.org/abs/1603.02188

[30] E. Androulaki and *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *EuroSys Conference, Porto, Portugal*, April 2018, pp. 30:1–30:15. [Online]. Available: http://doi.acm.org/10.1145/3190508.3190538

[31] B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 2, pp. 159–176, 1999. [Online]. Available: http://doi.acm.org/10.1145/317087.317089

[32] ——, "Cryptographic support for secure logs on untrusted machines," in *USENIX Security Symposium, San Antonio, USA*, Jan 1998. [Online]. Available: https://www.usenix.org/conference/7th-usenix-security-symposium/cryptographic-support-secure-logs-untrusted-machines

[33] B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an encrypted and searchable audit log," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2004, San Diego, California, USA*, 2004. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/04/proceedings/Papers/Waters.pdf

[34] A. A. Yavuz and P. Ning, "BAF: an efficient publicly verifiable secure audit logging scheme for distributed systems," in *Annual Computer Security Applications Conference, ACSAC, Honolulu, Hawaii, USA*, Dec 2009, pp. 219–228. [Online]. Available: https://doi.org/10.1109/ACSAC.2009.28

[35] D. Ma and G. Tsudik, "A new approach to secure logging," *TOS*, vol. 5, no. 1, pp. 2:1–2:21, 2009. [Online]. Available: http://doi.acm.org/10.1145/1502777.1502779

[36] D. Xu, L. Xiao, L. Sun, and M. Lei, "Game theoretic study on blockchain based secure edge networks," in *International Conference on Communications in China, ICCC, Qingdao, China*, Oct 2017, pp. 1–5. [Online]. Available: https://doi.org/10.1109/ICCChina.2017.8330529

[37] A. Sutton and R. Samavi, "Blockchain enabled privacy audit logs," in *International Semantic Web Conference ISWC, Vienna, Austria*, Oct 2017, pp. 645–660. [Online]. Available: https://doi.org/10.1007/978-3-319-68288-4 38

[38] L. Castaldo and V. Cinque, "Blockchain-based logging for the cross-border exchange of ehealth data in europe," in *Security in Computer and Information Sciences*, E. Gelenbe, P. Campegiani, T. Czachorski, S. K. Katsikas, I. Komnios, L. Romano, and D. Tzovaras, Eds.´ Cham: Springer International Publishing, 2018, pp. 46–56.

[39] J. Cucurull and J. Puiggali, "Distributed immutabilization of secure logs," in *International Workshop on Security and Trust Management STM Heraklion, Greece*, Sept 2016, pp. 122–137. [Online]. Available: https://doi.org/10.1007/978-3-319-46598-2 9

[40] S. Cha and K. Yeh, "An ISO/IEC 15408-2 compliant security auditing system with blockchain technology," in *2018 IEEE Conference on Communications and Network Security, CNS 2018, Beijing, China, May 30 - June 1, 2018*, 2018, pp. 1–2. [Online]. Available: https://doi.org/10.1109/CNS.2018.8433185

[41] J. Chen, S. Yao, Q. Yuan, K. He, S. Ji, and R. Du, "Certchain: Public and efficient certificate audit based on blockchain for TLS connections," in *IEEE Conference on Computer Communications, INFOCOM, Honolulu, HI, USA*, April 2018, pp. 2060–2068. [Online]. Available: https://doi.org/10.1109/INFOCOM.2018.8486344

[42] C. Wee, "Audit logs: to keep or not to keep?" in Recent Advances in Intrusion Detection, 1999. [Online]. Available: http://www.raid-symposium.org/raid99/PAPERS/Wee.pdf

[43] W. Olivier and R. von Solms, "The effective utilization of audit logs in information security management," in Annual Working Conference on Information Security Management & Small Systems Security, Sept 1999, pp. 51–62.

[44] C. Ringelstein and S. Staab, "DIALOG: distributed auditing logs," in IEEE International Conference on Web Services, ICWS, Los Angeles, USA, July 2009, pp. 429–436. [Online]. Available: https://doi.org/10.1109/ICWS.2009.50

[45] G. Baruffaldi and H. Sternberg, "Chains in chains - logic and challenges of blockchains in supply chains," in 51st Hawaii International Conference on System Sciences (HICSS), Hilton Waikoloa Village, Hawaii, USA, Jan 2018. [Online]. Available: http://aisel.aisnet.org/hicss-51/in/digital_supply_chain/3

[46] K. Fan, Y. Ren, Y. Wang, H. Li, and Y. Yang, "Blockchain-based efficient privacy preserving and data sharing scheme of content-centric network in 5g," IET Communications, vol. 12, no. 5, pp. 527–532, 2018. [Online]. Available: https://doi.org/10.1049/iet-com.2017.0619

[47] G. Danezis and S. Meiklejohn, "Centrally banked cryptocurrencies," in Proceedings of the 2016 Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, Feb. 2016. [Online]. Available: http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/centrally-banked-cryptocurrencies.pdf

[48] L. Mauri, S. Cimato, and E. Damiani, "A comparative analysis of current cryptocurrencies," Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP , Funchal, Madeira - Portugal, Jan. 2018, pp. 127–138. [Online]. Available: https://doi.org/10.5220/0006648801270138

[49] N. Stifter, A. Judmayer, P. Schindler, A. Zamyatin, and E. R. Weippl, "Agreement with satoshi - on the formalization of nakamoto consensus," IACR Cryptology ePrint Archive, vol. 2018, p. 400, 2018. [Online]. Available: https://eprint.iacr.org/2018/400

[50] A. E. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in Proceedings of the 37th IEEE Symposium on Security and Privacy (Oakland), San Jose, CA, May 2016, pp. 839–858. [Online]. Available: https://doi.org/10.1109/SP.2016.55

[51] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, N. Kulatova, A. Rastogi, T. Sibut-Pinote, N. Swamy, and S. Z. Béguelin, "Formal verification of smart contracts: Short paper," in Proceedings of the 23rd ACM Conference on Computer and Communications Security (CCS), Vienna, Austria, Oct. 2016, pp. 91–96. [Online]. Available: http://doi.acm.org/10.1145/2993600.2993611

[52] P. K. Sharma, S. Rathore, and J. H. Park, "Distarch-scnet: Blockchain-based distributed architecture with li-fi communication for a scalable smart city network," IEEE Consumer Electronics Magazine, vol. 7, no. 4, pp. 55–64, 2018. [Online]. Available: https://doi.org/10.1109/MCE.2018.2816745

[53] R. Guo, H. Shi, Q. Zhao, and D. Zheng, "Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems," IEEE Access, vol. 6, pp. 11 676–11 686, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2801266

[54] D. Rakic, "Blockchain technology in healthcare," in Proceedings of the 4th International Conference on Information and Communication Technologies for Ageing Well and e-Health, Funchal, Madeira, Portugal, March 2018., 2018, pp. 13–20. [Online]. Available: https://doi.org/10.5220/0006531600130020

[55] E. F. Jesus, V. R. L. Chicarino, C. V. N. de Albuquerque, and A. A. de A. Rocha, "A survey of how to use blockchain to secure internet of things and the stalker attack," Security and

Communication Networks, vol. 2018, pp. 9 675 050:1–9 675 050:27, 2018. [Online]. Available: https://doi.org/10.1155/2018/9675050

[56] I. Makhdoom, M. Abolhasan, H. Abbas, and W. Ni, "Blockchain's adoption in iot: The challenges, and a way forward," Journal of Network and Computer Applications, vol. 125, pp. 251 – 279, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804518303473

[57] E. Heilman, F. Baldimtsi, and S. Goldberg, "Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions," in Financial Cryptography and Data Security - International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, Feb 2016,, 2016, pp. 43–60. [Online]. Available: https://doi.org/10.1007/978-3-662-53357-4_4

[58] G. G. Dagher, P. B. Marella, M. Milojkovic, and J. Mohler, "Broncovote: Secure voting system using ethereum's blockchain," in Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP, Funchal, Madeira - Portugal, Jan 2018, pp. 96–107. [Online]. Available: https://doi.org/10.5220/0006609700960107

[59] F. S. Hardwick, R. N. Akram, and K. Markantonakis, "E-voting with blockchain: An e-voting protocol with decentralisation and voter privacy," CoRR, vol. abs/1805.10258, 2018. [Online]. Available: http://arxiv.org/abs/1805.10258

[60] M. M. Eljazzar, M. A. Amr, S. S. Kassem, and M. Ezzat, "Merging supply chain and blockchain technologies," Computing Research Repository (CoRR), vol. abs/1804.04149, 2018. [Online]. Available: https://goo.gl/5wMVJS

[61] M. Zhang and Y. Ji, "Blockchain for healthcare records: A data perspective," PeerJ PrePrints, vol. 6, p. e26942, 2018. [Online]. Available: https://doi.org/10.7287/peerj.preprints.26942v1

[62] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in 18th IEEE International Conference on e-Health Networking, Applications and Services, Munich, Germany, Sep 2016, pp. 1–3. [Online]. Available: https://doi.org/10.1109/HealthCom.2016.7749510

[63] A. Ahmad, M. Saad, M. Bassiouni, and A. Mohaisen, "Towards blockchain-driven, secure and transparent audit logs," in ACM International Workshop on Distributed Ledger of Things(DLoT), in conjunction with International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous, New York City, NY, USA, Nov 2018, pp. 443–448. [Online]. Available: https://doi.org/10.1145/3286978.3286985

[64] M. Vieira and H. Madeira, "Benchmarking the dependability of different OLTP systems," in International Conference on Dependable Systems and Networks (DSN), June 2003, pp. 305–310. [Online]. Available: https://doi.org/10.1109/DSN.2003.1209940

[65] C. Nikolaou, M. Marazakis, and G. Georgiannakis, "Transaction routing for distributed OLTP systems: Survey and recent results," Inf. Sci., no. 1&2, pp. 45–82, 1997. [Online]. Available: https://doi.org/10.1016/S0020-0255(96)00173-9

[66] U. Sirin, A. Yasin, and A. Ailamaki, "A methodology for OLTP micro-architectural analysis," in International Workshop on Data Management on New Hardware, DaMoN, Chicago, IL. ACM, May 2017, pp. 1:1–1:10. [Online]. Available: https://doi.org/10.1145/3076113.3076116

[67] W. Zhao, L. Qiang, H. Zou, A. Zhang, and J. Li, "Privacy-preserving and unforgeable searchable encrypted audit logs for cloud storage," in International Conference on Cyber

Security and Cloud Computing, CSCloud, Shanghai, China, M. Qiu, Ed. IEEE, June 2018, pp. 29–34. [Online]. Available: https://doi.org/10.1109/CSCloud/EdgeCom.2018.00015
[68] G. Hartung, "Secure audit logs with verifiable excerpts," in The Cryptographers' Track at the RSA Conference, San Francisco, CA, USA, ser. Lecture Notes in Computer Science, K. Sako, Ed., vol. 9610. Springer, Feb 2016, pp. 183–199. [Online]. Available: https://doi.org/10.1007/978-3-319-29485-8_11
[69] K. H. Lee, X. Zhang, and D. Xu, "Loggc: garbage collecting audit log," in ACM SIGSAC Conference on Computer and Communications Security CCS, Berlin, Germany, Nov 2013, pp. 1005–1016. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516731
[70] J. Margulies, "A developer's guide to audit logging," IEEE Security & Privacy, vol. 13, no. 3, pp. 84–86, 2015. [Online]. Available: https://doi.org/10.1109/MSP.2015.50
[71] R. Luh, S. Marschalek, M. Kaiser, H. Janicke, and S. Schrittwieser, "Semantics-aware detection of targeted attacks: a survey," J. Computer Virology and Hacking Techniques, vol. 13, no. 1, pp. 47–85, 2017. [Online]. Available: https://doi.org/10.1007/s11416-016-0273-3
[72] B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics," ACM Trans. Inf. Syst. Secur, vol. 2, no. 2, pp. 159–176, 1999. [Online]. Available: https://goo.gl/psvN2a
[73] "Cryptographic support for secure logs on untrusted machines," in USENIX Security Symposium, San Antonio, USA, Jan 1998. [Online]. Available: https://www.usenix.org/conference/7th-usenix-security-symposium/cryptographic-support-secure-logs-untrusted-machines
[74] R. Snodgrass, S. S. Yao, and C. Collberg, "Tamper detection in audit logs," in Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, Aug 2004, pp. 504–515. [Online]. Available: http://www.vldb.org/conf/2004/RS13P1.PDF
[75] I. Ray, K. Belyaev, M. Strizhov, D. Mulamba, and M. Rajaram, "Secure logging as a service—delegating log management to the cloud," IEEE systems journal, vol. 7, no. 2, pp. 323–334, 2013.
[76] D. Ma and G. Tsudik, "A new approach to secure logging," TOS, vol. 5, no. 1, pp. 2:1–2:21, 2009. [Online]. Available: https://doi.org/10.1145/1502777.1502779
[77] A. A. Yavuz, P. Ning, and M. K. Reiter, "BAF and FI-BAF: efficient and publicly verifiable cryptographic schemes for secure logging in resource-constrained systems," ACM Trans. Inf. Syst. Secur., vol. 15, no. 2, pp. 9:1–9:28, 2012. [Online]. Available: https://doi.org/10.1145/2240276.2240280
[78] H. Hyvärinen, M. Risius, and G. Friis, "A blockchain-based approach towards overcoming financial fraud in public sector services," Business & Information Systems Engineering, vol. 59, no. 6, pp. 441–456, 2017. [Online]. Available: https://doi.org/10.1007/s12599-017-0502-4
[79] F. Holotiuk, F. Pisani, and J. Moormann, "The impact of blockchain technology on business models in the payments industry," in Towards Thought Leadership in Digital Transformation: 13. Internationale Tagung Wirtschaftsinformatik, St.Gallen, Switzerland, Feb, 2017., 2017. [Online]. Available: http://aisel.aisnet.org/wi2017/track09/paper/6
[80] International Conference on Software Quality, Reliability and Security Companion, QRS Companion, Lisbon, Portugal. IEEE, 2018. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8424928
[81] D. Derler, K. Samelin, D. Slamanig, and C. Striecks, "Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based," IACR Cryptology ePrint Archive, vol. 2019, p. 406, 2019. [Online]. Available: https://eprint.iacr.org/2019/406

[82] A. Shahaab, B. Lidgey, C. Hewage, and I. Khan, "Applicability and appropriateness of distributed ledgers consensus protocols in public and private sectors: A systematic review," IEEE Access, vol. 7, pp. 43 622–43 636, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2904181

[83] X. Yi, T. Yang, J. Wu, and K. H. Johansson, "Distributed event-triggered control for global consensus of multi-agent systems with input saturation," Automatica, vol. 100, pp. 1–9, 2019. [Online]. Available: https://doi.org/10.1016/j.automatica.2018.10.032

[84] M. Wang, M. Duan, and J. Zhu, "Research on the security criteria of hash functions in the blockchain," in ACM Workshop on Blockchains, Cryptocurrencies, and Contracts, BCC-AsiaCCS, Incheon, Republic of Korea, S. V. Lokam, S. Ruj, and K. Sakurai, Eds. ACM, June 2018, pp. 47–55. [Online]. Available: https://doi.org/10.1145/3205230.3205238

[85] M. Saad, V. Cook, L. Nguyen, M. T. Thai, and A. Mohaisen, "Partitioning attacks on bitcoin: Colliding space, time, and logic," in IEEE International Conference on Distributed Computing Systems ICDCS, Dallas, Texas, US, July 2019.

[86] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Consensus in the age of blockchains," CoRR, vol. abs/1711.03936, 2017. [Online]. Available: http://arxiv.org/abs/1711.03936

[87] . Sutton and R. Samavi, "Blockchain enabled privacy audit logs," in International Semantic Web Conference ISWC, Vienna, Austria, Oct 2017, pp. 645–660. [Online]. Available: https://doi.org/10.1007/978-3-319-68288-4_38

[88] L. Castaldo and V. Cinque, "Blockchain-based logging for the cross-border exchange of ehealth data in europe," in Security in Computer and Information Sciences, E. Gelenbe, P. Campegiani, T. Czachórski, S. K. Katsikas, I. Komnios, L. Romano, and D. Tzovaras, Eds. Cham: Springer International Publishing, 2018, pp. 46–56.

[89] J. Cucurull and J. Puiggali, "Distributed immutabilization of secure logs," in International Workshop on Security and Trust Management STM Heraklion, Greece, Sept 2016, pp. 122–137. [Online]. Available: https://doi.org/10.1007/978-3-319-46598-2_9

[90] A. Alghamdi, M. S. Owda, and K. A. Crockett, "Natural language interface to relational database (NLI-RDB) through object relational mapping (ORM)," in Advances in Computational Intelligence Systems - Contributions Presented at the 16th UK Workshop on Computational Intelligence, Lancaster, UK, ser. Advances in Intelligent Systems and Computing, P. Angelov, A. E. Gegov, C. Jayne, and Q. Shen, Eds., vol. 513. Springer, Sept 2016, pp. 449–464. [Online]. Available: https://doi.org/10.1007/978-3-319-46562-3_29

[91] H. Bagheri, C. Tang, and K. J. Sullivan, "Automated synthesis and dynamic analysis of tradeoff spaces for object-relational mapping," IEEE Trans. Software Eng., vol. 43, no. 2, pp. 145–163, 2017. [Online]. Available: https://doi.org/10.1109/TSE.2016.2587646

[92] N. Community, "Nhibernate," 2018. [Online]. Available: http://nhibernate.info/

[93] A. J. Ganesh, A. Kermarrec, and L. Massoulié, "Peer-to-peer membership management for gossip-based protocols," IEEE Trans. Computers, vol. 52, no. 2, pp. 139–149, 2003. [Online]. Available: https://doi.org/10.1109/TC.2003.1176982

[94] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "SoK: Consensus in the Age of Blockchains," 2017, https://arxiv.org/abs/1711.03936.

[95] D. Crockford, "The application/json media type for javascript object notation (JSON)," RFC, vol. 4627, pp. 1–10, 2006. [Online]. Available: https://doi.org/10.17487/RFC4627

[96] J. Huai, R. Chen, H. Hon, Y. Liu, W. Ma, A. Tomkins, and X. Zhang, Eds., Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008. ACM, 2008.
[97] M. Saad and A. Mohaisen, "Towards characterizing blockchain-based cryptocurrencies for highly-accurate predictions," in IEEE Conference on Computer Communications Workshops, April 2018.
[98] G. Nguyen and K. Kim, "A survey about consensus algorithms used in blockchain," JIPS, vol. 14, no. 1, pp. 101–128, 2018. [Online]. Available: https://doi.org/10.3745/JIPS.01.0024
[99] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. E. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains," in International Workshop on Financial Cryptography and Data Security, Christ Church, Barbados, Feb 2016, pp. 106–125. [Online]. Available: https://doi.org/10.1007/978-3-662-53357-4_8
[100] M. Vukolic, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in International Workshop on Open Problems in Network Security - IFIP WG 11.4 , iNetSec, Zurich, Switzerland, ser. Lecture Notes in Computer Science, J. Camenisch and D. Kesdogan, Eds., vol. 9591. Springer, Nov 2015, pp. 112–125. [Online]. Available: https://doi.org/10.1007/978-3-319-39028-4_9
[101] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of PBFT consensus process for permissioned blockchain network (hyperledger fabric)," in 36th IEEE Symposium on Reliable Distributed Systems, SRDS 2017, Hong Kong, Hong Kong, September 26-29, 2017, 2017, pp. 253–255. [Online]. Available: https://doi.org/10.1109/SRDS.2017.36
[102] C. Cachin, "Architecture of the hyperledger blockchain fabric," in Workshop on Distributed Cryptocurrencies and Consensus Ledgers, vol. 310, 2016.
[103] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. W. Cocco, and J. Yellick, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in EuroSys Conference, Porto, Portugal, April 2018, pp. 30:1–30:15. [Online]. Available: http://doi.acm.org/10.1145/3190508.3190538
[104] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in Proceedings of the 38th IEEE Symposium on Security and Privacy (Oakland). San Jose, CA: IEEE, May 2017, pp. 375–392. [Online]. Available: https://doi.org/10.1109/SP.2017.29
[105] I. Abraham and D. Malkhi, "The blockchain consensus layer and BFT," Bulletin of the EATCS, vol. 123, 2017. [Online]. Available: http://eatcs.org/beatcs/index.php/beatcs/article/view/506
[106] J. Göbel and A. E. Krzesinski, "Increased block size and bitcoin blockchain dynamics," in 2017 27th International Telecommunication Networks and Applications Conference (ITNAC). IEEE, 2017, pp. 1–6.
[107] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer et al., "On scaling decentralized blockchains," in International Conference on Financial Cryptography and Data Security. Springer, 2016, pp. 106–125.

[108] S. D. Angelis, "Assessing security and performances of consensus algorithms for permissioned blockchains," CoRR, vol. abs/1805.03490, 2018. [Online]. Available: http://arxiv.org/abs/1805.03490
[109] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Consensus in the age of blockchains," CoRR, vol.
[110] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *IEEE Computer*, vol. 50, no. 7, pp. 80–84, 2017.
[111] M.-H. Maras, "Internet of Things: security and privacy implications," *International Data Privacy Law*, vol. 5, no. 2, pp. 99–104, May 2015. [Online]. Available: https://academic.oup.com/idpl/article/5/2/99/645234
[112] A. Dey, K. Stuart, and M. E. Tolentino, "Characterizing the impact of topology on iot stream processing," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, Feb 2018, pp. 505–510.
[113] M. Crosby, "BlockChain Technology: Beyond Bitcoin," *Applied Innovation Review (AIR)*, no. 2, p. 16, 2016.
[114] N. Kshetri, "Can blockchain strengthen the internet of things?" *IT Professional*, vol. 19, no. 4, pp. 68–72, 2017.
[115] N. Suri, M. Tortonesi, J. Michaelis, P. Budulas, G. Benincasa, S. Russell, C. Stefanelli, and R. Winkler, "Analyzing the Applicability of Internet of Things to the Battlefield Environment," May 2016
[116] D. K. Tosh, S. Shetty, P. Foytik, L. Njilla, and C. A. Kamhoua, "Blockchain-Empowered Secure Internet -of- Battlefield Things (IoBT) Architecture," in *IEEE Military Communications Conference (MILCOM)*, Oct. 2018, pp. 593–598.
[117] G. Padmavathi and D. Shanmugapriya, "A survey of attacks, security mechanisms, and challenges in wireless sensor networks," *International Journal of Computer Science and Information Security*, vol. 4, no. 1, 2009.
[118] R. Sepe, "Denial of Service Deterrence," *SANS Institute Information Security Reading Room*, p. 25, 2015
[119] P. Sass, "Communications networks for the force xxi digitized battle- field," *Springer US Mobile Networks and Applications*, pp. 139–155, 1999.
[120] "Four future trends In tactical network modernization." https://www.army.mil/article/216031/four future trends in tactical network modernization. Last Accessed: May 31, 2019.
[121] S. Misra, S. K. Dhurandher, A. Rayankula, and D. Agrawal, "Using hon- eynodes for defense against jamming attacks in wireless infrastructure- based networks," *Computers & Electrical Engineering*, vol. 36, pp. 367– 382, Mar. 2010.
[122] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, pp. 557–564, IEEE, June 2017. event-place: Honolulu, HI, USA.
[123] L. Yushi, J. Fei, and Y. Hui, "Study on application modes of Military Internet of Things (MIoT)," in *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, vol. 3, pp. 630–634, IEEE, 2012.
[124] M. J. Farooq and Q. Zhu, "Secure and reconfigurable network design for critical information dissemination in the internet of battlefield things (iobt)," in *IEEE Intl.*

Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), pp. 1–8, 2017.
[125] N. Abuzainab and W. Saad, "Dynamic connectivity game for adversarial internet of battlefield things systems," *IEEE Internet of Things Journal*, 2017.
[126] M. Tortonesi, A. Morelli, M. Govoni, J. Michaelis, N. Suri, C. Stefanelli, and S. Russell, "Leveraging internet of things within the military network environment x2014; challenges and solutions," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 111–116, Dec 2016.
[127] P. P. Ray, "Towards an internet of things based architectural framework for defense," in *2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pp. 411–416, Dec 2015.
[128] A. Sanjab, W. Saad, and T. Basar, "Prospect theory for enhanced cyber- physical security of drone delivery systems: A network interdiction game," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2017.
[129] O. Bello and S. Zeadally, "Intelligent device-to-device communication in the internet of things," *IEEE Systems Journal*, vol. 10, no. 3, pp. 1172–1182, 2016.
[130] C. Law and K.-Y. Siu, "Distributed construction of random expander networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3, pp. 2133–2143, IEEE, 2003.
[131] H. Bauer, M. Patel, and J. Veira, "Internet of Things: Opportunities and challenges for semiconductor companies." https: //www.mckinsey.com/industries/semiconductors/our-insights/internet- of- things- opportunities- and- challenges- for- semiconductor- companies.
[132] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.
[133] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.
[134] L. Lamport *et al.*, "Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.
[135] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof- of-stake," *self-published paper, August 19, 2012*, 2012.
[136] D. Tosh, S. Shetty, P. Foytik, C. Kamhoua, and L. Njilla, "Cloudpos: A proof-of-stake consensus design for blockchain integrated cloud," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 302–309, IEEE, 2018.
[137] "Introduction — Sawtooth v1.1.4 documentation." https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html#. Last Accessed: May 31, 2019.
[138] J. Ahrenholz, "CORE Documentation." https://www.nrl.navy.mil/itd/ncs/products/core. Last Accessed: May 31, 2019.
[139] D. Mazieres, "The stellar consensus protocol: A federated model for internet-level consensus." Stellar Development Foundation, 2015.
[140] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 51–68, 2017.
[141] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, *et al.*, "On scaling decentralized blockchains," in *International Conference on Financial Cryptography and Data Security*, pp. 106–125, Springer, 2016.

7 LIST OF ACRONYMS

[1] POW – Proof of Work
[2] IoBT – Internet of Battlefield Things
[3] PBFT – Practical Byzantine Fault Tolerance
[4] IoT – Internet of Things
[5] JSON – Javascript Object Notation
[6] ORM – Object Relational Mapping
[7] CORE – Common Open Research Emulator