

© 2023 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Kamal, Mohsin, Muhammad Tariq, Mian Ahmad Jan, and Houbing Song. "Blockchain Enabled Secure Distributed Event Logging in the Industrial Internet of Things." IEEE Internet of Things Journal, 2023, 1–1. <https://doi.org/10.1109/JIOT.2023.3343622>.

<https://doi.org/10.1109/JIOT.2023.3343622>

Access to this work was provided by the University of Maryland, Baltimore County (UMBC) ScholarWorks@UMBC digital repository on the Maryland Shared Open Access (MD-SOAR) platform.

**Please provide feedback**

Please support the ScholarWorks@UMBC repository by emailing [scholarworks-group@umbc.edu](mailto:scholarworks-group@umbc.edu) and telling us what having access to this work means to you and why it's important to you. Thank you.

# Blockchain Enabled Secure Distributed Event Logging in the Industrial Internet of Things

Mohsin Kamal, *Senior Member, IEEE*, Muhammad Tariq, *Senior Member, IEEE*, Mian Ahmad Jan, *Senior Member, IEEE*, Houbing Song, *Fellow, IEEE*

**Abstract**—Blockchain technology has found applications across diverse domains owing to its ability to establish trust in a decentralized manner. Nevertheless, the integration of blockchain into critical infrastructure domains encounters significant challenges posed by the computational demands and storage requirements associated with the proof-of-work puzzle during the mining process. This scenario becomes particularly complex in the context of applications within the Industrial Internet of Things (IIoT), where stringent timeliness constraints are inherent, notably in functions such as intrusion detection and control. This paper presents a novel solution that takes into account the time-sensitive nature of application constraints within the IIoT. Specifically, we focus on online functions involving intrusion detection and control. By doing so, we address the imperative need for timely and secure data delivery, crucial in maintaining the integrity of hard-to-tamper ledger blocks. These blocks encapsulate measurements that are seamlessly utilized by various system functions and components. The proposed approach optimizes the utilization of heterogeneous resources governing blockchain computations. This optimization ensures that the desired properties for logging within the blockchain are met, enabling the prompt delivery of measurements. The novel collaborative mining technique entails the sharing of nonce ranges among miners, which effectively reduces the overall mining time and enhances the efficiency of the process.

**Index Terms**—Blockchain, Industrial Internet of Things, Edge Computing, Event Logging, Proof of Work.

## I. INTRODUCTION

The Industrial Internet of Things (IIoT) utilizes electronic devices and digital communication to efficiently monitor and control industrial processes remotely. Its driving forces are inherent sensing and actuating components. These sensors collect environmental data, sent

across networks to remote control centers. The data is analyzed by control devices, issuing precise commands to actuating components, ensuring seamless industrial functionality. IIoT systems are often integrated into critical industrial environments like manufacturing plants and energy facilities. They play a key role in ensuring smooth operations. However, due to their extensive integration, they become attractive targets for malicious activities driven by economic or political motives.

Maintaining the precise *audit logs* of the events is very crucial in the context of IIoT to tackle the potential vulnerabilities in the design and operations of industrial systems [1]. These events include the up-loading measurements, executing commands and login access to various devices. The primary objective of these audit logs are to support real-time functions for intrusion detection systems [2]. Furthermore, they play a vital role in data forensic [3] and identification of any misuse of the system carried out by the adversary. Typically, the responsibility of generating these logs falls on Supervisory Control and Data Acquisition (SCADA) components, which also undertake the task of estimating the system's operational state. Audit logs proved crucial during a cyber attack on the Ukrainian power grid [3]. Investigators used them to analyze the attackers' actions, although some logs were intentionally deleted. This highlights the need for comprehensive and tamper-resistant audit logs to enhance industrial system security against cyber threats.

For data integrity, robust secure logging mechanisms are vital. These should prevent fake log entries, unauthorized modifications, and deletions. In IIoT, preserving audit log integrity is crucial for security and accountability. An important element is *forward integrity*, which shields pre-compromise log data from tampering without detection. The authors in [4] introduce *Forward-Secure Sequential Aggregate (FssAgg)* authentication which uses public-private key pairs to sign new log entries at intervals. Within each interval, event signatures are sequentially aggregated into one, replacing individual signatures. This aggregate signature validates the whole log during verification. Failure indicates an invalid signature. However, the authors in [4] and [5] don't

Manuscript received on Aug. 16, 2023. In revised form on Oct. 8, and Nov. 06, 2023. Accepted on Nov. 29, 2023

Mohsin Kamal is with the School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Pakistan (E-mail: dr.mohsinkamal@seecs.edu.pk)

Muhammad Tariq (Corresponding author) is with the Department of Electrical Engineering, National University of Computer and Emerging Sciences, Pakistan (E-mail: mtariq@princeton.edu).

Mian Ahmad Jan is with the Department of Computer Science, College of Computing and Informatics, University of Sharjah, Sharjah, 27272, UAE (Email: mjan@sharjah.ac.ae)

Houbing Song is with the University of Maryland, Baltimore County, USA (Email: h.song@ieee.org)

cover ensuring logging service availability. This matters in IIoT, where continuous operation is vital, requiring attention in system design.

*Distributed event logging* resolves limitations in centralized methods such as [4] and [5]. This improves audit log availability by distributing and replicating logs across devices, even in different locations. However, this brings new challenges in maintaining replica consistency due to replication. Ensuring consistency amid concurrent changes or tampering, especially in chronological order, is crucial for intrusion detection systems [6]. To attain this consistency in a distributed setting, consensus, as discussed in [7], can be applied. Consensus mechanisms allow replicas to mutually agree on log content and order. However, achieving consensus is intricate due to challenges like infeasibility in asynchronous systems with single crash failure [8], scalability concerns [9], and substantial communication overhead [7]. A more intuitive distributed structure for ordered data is a *distributed ledger* [10], commonly known as a *blockchain* [11]. A blockchain is a growing list of *blocks*, each linked to the prior one via a cryptographic hash, maintaining order. Each block typically holds data with ordering requirements; for audit logs, this could mean log entries. To establish block order, miners (replicas) use a probabilistic consensus protocol in blockchain. Miners aim to solve computational puzzles, agreeing on the first solved puzzle's proposed block. These puzzles demand substantial computational power, leading to Proof-of-Work (PoW) as proof of puzzle completion. Beyond consensus, PoW secures blocks, deterring malicious attempts to alter incorporated blocks. The computation demands of the PoW, pose a major challenge for the adaption of blockchain into IIoT for secure event logging. On the other hand, IIoT operates under tolerable delay, i.e., an amount of delay that can still preserve smooth operation. Thus, sensor data derived from the physical world must be propagated in a timely and reliable manner to be used for online computations. One way to indicate the acceptable latency for each event in the IIoT is to have a timestamp attribute with the physical time to be delivered and meet the specified time deadline.

Key contributions of this paper include:

- addressing the security challenge of logging sensor events into blockchain with controlled mining parameters, crucial for online functions,
- introducing a novel collaborative mining approach where miners share attributes to reduce operational costs, and
- implementing distributed clustering of miners to enhance availability and mitigate single points of failure.

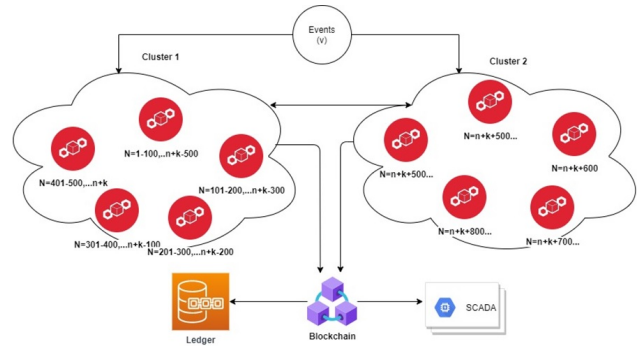


Fig. 1: Proposed blockchain architecture as envisioned to be integrated in the IIoT.

## II. MODEL OF COMPUTATION

### A. System model

The system model is presented in Fig. 1. Our goal is to record sensing measurements in a blockchain data structure for increased integrity, availability, and reliability. A *blockchain*  $B$  is a list of blocks where each block  $b_i$ , gets a unique identifier from a set  $i \in I$ . A block  $b_i$  is composed by a subset of events in  $e \in E$ , denoted as  $b_i.E$ .

We consider a system that consists of three sets of computing devices, we refer as nodes: (i) a set of sensor nodes  $V = \{v_1, \dots, v_c\}$  to generate the set of events  $E$ , (ii) a set of intrusion detection nodes (e.g. SCADA components)  $U = \{u_1, \dots, u_l\}$ , and (iii) a set of nodes residing in the cluster  $M$ , named miners  $M_i = \{m_1, \dots, m_j\}$ . We consider miners to have heterogeneous computational capabilities where there exist a set of  $K$  different types of miners. Each category  $k$  within the set  $K$  is defined by a parameter tuple  $\langle A_k, \Gamma_k, c_k \rangle$ . Here,  $A_k$  signifies the computational capacity, measured in hash trials per second. Consider  $A_1$  as the computational rate of the least performing miner category; then,  $\Gamma_k = \frac{A_k}{A_1}$  illustrates the relative speed at which type  $k$  miners operate in comparison to the slowest category. The measure of *operational efficiency* (watt/hash trials) is denoted by  $c_k$ . We assume that the nodes communicate via messages, through *reliable* communication channels. That is, if a node  $n_i \in V \cup U \cup M$  sends a message  $m$  to a node  $n' \in V \cup U \cup M$ , then  $n'$  eventually receives  $m$ . We assume that channels are subject to a bounded capacity. Let,  $R_{mu}$  denote the channel data rate between miners and IDS nodes, and  $R_{mm}$  the data rate of the channels between the miners, where, both rates are measures in  $\frac{bits}{sec}$ .

We use a global clock  $C$  to examine the progress of the system. Each node  $n \in V \cup U \cup M$  has access to a local clock  $c_i$  which is loosely synchronized to  $C$ . That

is, at any instant, the local clock  $c_n \in [C - \zeta, C + \zeta]$ , with bounded error  $\zeta$  on the notion of time. Such setup is quite practical via time generic clock synchronization protocols like NTP [12] or using techniques such as IEEE 1588 [13] which delivers a striking precision.

Let sensor  $v_n \in V$ , produces an event  $e = v_n(t)$  at time  $t$ . The event  $e$  is transmitted instantaneously to all the clusters  $M_i$   $\forall i = 1, 2, \dots, n$ , before it gets communicated among all miners in  $M_i$  residing in the cluster. Once a block  $b_i$ , and its events  $b_i.\mathcal{E} \subseteq E$ , is appended in  $B$ , is also sent to a subset of IDS nodes,  $U' \subseteq U$ . An event header is a triple of the form  $\langle t, m, R' \rangle_e$  where:

- $t_e$ : the time to live (TTL)
- $m_e$ : the event size
- $R'$ : the data rate of the channel between miners and the IDS nodes,  $\forall u \in U$

The TTL field indicates the maximum latency to consider the measurement valid from the time of creation, with respect to  $C$ . The size  $m_e$  for event  $e$  (including the header and sensor reading sizes) is a real number measured in bits. Lastly, the rate  $R_e$ , denotes the channel capacity between the miners and any node  $u \in U$ , and is measured in  $\frac{\text{bits}}{\text{sec}}$ . Similarly, the data rate of channels between miners, and the clusters are  $R_{m_i, m_j}$  and  $R_{M_i, M_j}$ , respectively. Its not until an event is included in a block and delivered to  $u \in U'$  that becomes available for online processing, the total event delay noted with  $D_e$  is

$$D_e = w(\log(|M|) + 3\log(\log(|M|))) + \theta_i + \frac{Z_i}{R_e} + \zeta \quad (1)$$

where event  $e$  is re-transmitted for  $\log(|M|) + 3\log(\log(|M|))$  times among a set of miners in all clusters  $M$  and each re-transmission takes  $w$  seconds. The available time (in seconds) an event has in order to be included in block  $b_i$  is  $\theta_i$ , where it depends on the combination of measurements it includes,  $b_i.\mathcal{E}$ . The propagation of time of block of size  $Z_i$  to  $u \in U'$  is limited by the finite channel bitrate  $R_e$  and is  $\frac{Z_i}{R_e}$ . The term  $\zeta$  is the maximum synchronization delay of local clock with respect to  $C$ .

In order to ensure proper smooth operation of the IIoT the proposed blockchain model must ensure the two following safety properties:

- **P1:**  $\forall e \in E, \exists b_{i\tau\xi}$  s.t.  $e \in b_{i\tau\xi}.\mathcal{E}$
- **P2:**  $\forall e \in b_{i\tau\xi}.\mathcal{E}, \exists D_e$  s.t.  $D_e \leq t_e$

The first property is for every event  $e \in E$  that is transmitted from sensors must be included at least in one of validated block  $b_{i\tau\xi}.\mathcal{E}$ . It holds as long as  $e$  is transmitted to a non compromised miner. The second property is for every event  $e \in b_{i\tau\xi}.\mathcal{E}$  to have  $D_e$  delay that is not greater than  $y_e$ , the indicated event's TTL, in

order to be used by the intended online functions before it becomes obsolete.

## B. PoW Agreement

Blocks may be generated concurrently by many processes. Its essential to create a mechanism that will insert these blocks in a consistent order in blockchain. We assume Proof of Work (PoW) consensus mechanism, for validating the appended blocks and agreeing on their ordering. The validation process based on PoW is termed mining, while miners involved in this process are thought as solving a computationally hard puzzle (PoW puzzle). In a nutshell, it involves iteratively computing the output of a cryptographic hash function ( $H : \{0, 1\}^\alpha \rightarrow \{0, 1\}^\beta$ ) having as input the data of block header  $b_i.h$ , along with a counter field termed *nonce*. At each iteration the *nonce* increases within a search space of size  $r$  until a value of the *nonce* is found to validate  $b_i.h$  by satisfying with the following condition:

$$H(b_i.h || \text{nonce}) \leq N \quad (2)$$

where  $N : \{0, 1\}^n$  is a number with  $d$  most-significant bits to be zero. Parameter  $d$  is part of the block header and its thought as the difficulty level of the computational process. Moreover, field *prev* in the  $b_i.h$  is a reference pointing to the previous block in the header. Finally, the variable  $x_{ie} \in \{0, 1\}$  indicates that event  $e \in E$  is included in block  $b_i$  when  $x_{ie} = 1$  and otherwise is  $x_{ie} = 0$ .

In this setting the miners are authorized nodes which operate on predefined rules for computing  $b_i$ . Having control over operation of miners enables parameters in  $B$  to be tuned for optimizing operational performance. The amount of type  $k$  miners assigned to distributively solve the PoW puzzle for  $b_i$  is  $s_{ik}$ .

In practicality, uncertainty manifests in the count of received events and their characteristics, such as Time-to-Live (TTL). To properly allocate miners for processing new events within blocks, forecasts regarding event arrivals are generated. This necessitates the creation of a scenario tree, akin to the methodology detailed in [14]. The tree facilitates the arrangement of events into blocks, involving a set of  $|\Xi|$  realized scenarios, denoted by  $\xi \in \Xi$ . Each scenario encompasses  $T$  successive stages (referred to as nodes), with stage index  $\tau = 1, 2, \dots, T$  denoting the depth from the root node. The stochastic variables in question encompass (i) the volume of events entering the blockchain in batches, denoted as  $\Lambda_{\tau\xi}$ , and (ii) their respective TTL values, denoted as  $y_{e\tau\xi}$ . For each amalgamation of stage and scenario, a probability  $\rho_{\tau\xi}$  exists, wherein  $\sum_{\tau \in T} \rho_{\tau\xi} = 1$ , aligning with the

probability of specific uncertain parameters manifesting within a given scenario. Evidently, the size of the scenario tree escalates exponentially as parameters and stages expand.

Let  $\mu = (1 - 2^{-d})$  as defined in [15], denoting the likelihood of successfully discovering block  $i \in I$  within a single attempt. Additionally, consider  $\phi_k = \Gamma_k m_{ik\tau\xi}$  and  $\phi = \sum_{k \in K} \Gamma_k m_{ik\tau\xi}$ , representing the trials undertaken by a single miner type and multiple miner types, respectively. The probability of miners effectively unearthing a valid block subsequent to  $\phi\sigma$  unsuccessful trials can be expressed as

$$p_\sigma = \mu^{\phi\sigma} (1 - \mu^\phi) \quad (3)$$

Setting  $\sigma = A_1 \theta_{i\tau\xi}$ , then  $p_{A_1 \theta_{i\tau\xi}}$  expresses the probability of failures within time interval  $\theta_{i\tau\xi}$  before finding a valid block.

Therefore, for difficulty level  $d$ , the probability miners fail in generating a block within time interval  $\theta$  is bounded above by  $\delta$  as

$$P_{m, \theta_{i\tau\xi}} = 1 - \sum_{\sigma=0}^{\theta_{i\tau\xi} A_1} \mu^{\phi\sigma} (1 - \mu^\phi) = 1 - \mu^{\phi(\theta_{i\tau\xi} A_1 + 1)} \quad (4)$$

where it is obtained from the cumulative probability of succeeding within time interval  $\theta$ .

Moreover, the event of a successfully mined block becomes invalid due to being part of a fork is called orphaning. The occurrence of such an event depends both on the block size which impacts  $\theta'$ , the propagation delay in the blockchain network, and difficulty level  $d$ . The probability  $P'_{s, \theta'}$  for a block to become an orphan is derived by subtracting from  $P_{s, \theta}$  the event where only a single miner finds the PoW block within  $\theta'$  seconds as

$$P'_{m, \theta'_{i\tau\xi}} = P_{m, \theta'_{i\tau\xi}} - \phi \mu^{\theta'_{i\tau\xi} A_1 (\phi-1)} (1 - \mu^{\theta'_{i\tau\xi} A_1}) \quad (5)$$

Thus, the larger the block size, the longer the latency, and will be more likely to become part of a fork. We consider the parameters  $(d, \delta, \delta_2)$  to be constant for consecutive blocks and defined explicitly by system operators.

### III. HEURISTICS AND ALGORITHM

The primary goal for implementing blockchain-based systems is to achieve data integrity with low resource utilization. The system model in Fig. 1 presents a clustering-based approach in which the resources are shared among the miners which helps in reducing the mining time ( $T_m$ ) for  $e_i$  in  $b_i$ . The events are generated and pre-processed initially, which are sent to  $m_i$  for mining in  $M_j$ . The mining process can be collaborative

or competitive, based on the TTL associated with the events. After mining,  $b_i$  is added to the  $B$ , and the events are sent to SCADA or any other systems for further operations. Furthermore, the hash ( $H$ ) of the block is also added to the ledger for future authentication and forensic operations. Initially, based on the capabilities of miners, dummy events are mined to achieve the mean mining time ( $\mu_{T_m}$ ) and standard deviation of the mining time ( $\sigma_{T_m}$ ). Based on these values, a threshold ( $Th$ ) is computed as

$$Th = \mu_{T_m} + \sigma_{T_m} + p_{delay} \quad (6)$$

where  $p_{delay}$  is the propagation delay which depends on the payload size ( $P$ ) of the event and bandwidth ( $BW$ )

$$p_{delay} = \frac{P}{BW} \quad (7)$$

The detailed insight of each entity in the model is given as follows.

1) *Transaction Node*: At the transaction node, the events are generated randomly where each event is associated with a TTL. These events are comprised of different payload sizes. The events are sent to the scheduler node for further processing. The transaction node is shown in Algorithm 1, line 8.

2) *Scheduler Node*: The main objective of the scheduler node is to arrange events in blocks. The first step is to sort events in ascending order according to their TTL. The benchmark of previous  $Th$  is used in current operations in which the TTL values of all events are compared with the  $Th$ . If the value of TTL is below  $Th$ , then the event associated with that TTL is added to the list of expired events. The TTL values are segmented into five groups, for example, based on the maximum TTL values that our system can achieve, the 0-20% of the TTL is denoted as group 1, and so on. In total we get five groups for different levels of TTL. The non-expired events are distributed in these groups depending on the value of their respective TTL.

The maximum TTL,  $P$ , and  $e$  are categorized as Low (L), Medium (M), and High (H) by distributing the overall values into three equally sized ranges. Based on the information of mean TTL ( $\mu^{TTL}$ ),  $\mu^P$  and  $\mu^e$ , the decision about the collaborative and competitive mining is taken by the scheduler node and the information is communicated to all the clusters.

$$\mu^{\{TTL, P, e\}} = \frac{\sum_{i=0}^{n-1} \{TTL_i, P_i, e_i\}}{e \times \{\max(TTL), \max(P), 1\}} \quad \forall E \quad (8)$$

The primary focus is on prioritizing Low  $\mu^{TTL}$  values, regardless of  $\mu^P$  and  $\mu^e$ , in order to maximize the number of events mined and included in the blocks. Consequently, the collaborative mining approach is initiated. For medium and high  $\mu^{TTL}$  values, the competitive

**Algorithm 1:** Algorithm to mine blocks using TTL information of the events.

---

**input :** Events data  
**output:** Blocks generation

- 1 **Threshold\_Update**(Payload size,
- 2  $MiningTime_{mean}, \sigma$ );
- 3     Propagation delay =
- 4     Payload size/bandwidth;
- 5     Threshold =
- 6      $MiningTime_{mean} + \sigma + \text{Propagation delay}$ ;
- 7 **Combiner**(events);
- 8     Group events w.r.t TTL;
- 9     Return batch of events;
- 10 **// Transaction Node:**
- 11 Events from Transaction node are sent to Scheduler node;
- 12 **// Scheduler Node:**
- 13 The events are sorted with respect to the TTL;
- 14 **if** TTL < Threshold **then**
- 15     Add event to the list of expired events pool;
- 16     **else** Block = Combiner(events);
- 17 **Based on previous Payload, TTL and events,**
- 18 **do;**
- 19 **if** TTL is Low **then**
- 20     Miner\_Collaborative(block);
- 21 **else if**
- 22      $(T1 \& P0 \& E0) \text{ or } (T2 \& P0 \& E0) \text{ or } (T2 \& P0 \& E1)$
- 23      $\text{or } (T2 \& P1 \& E0)$  **then**
- 24     Miner\_competitive(block);
- 25     **else** Miner\_Collaborative(block);
- 26     ;
- 27 **// Miner Node:**
- 28 **Miner\_Collaborative(block);**
- 29     Assign upper and lower bounds of nonces to miners;
- 30 **for** all miners having UB and LB of nonce values **do**
- 31     **if** block is mined **then**
- 32     Break;
- 33     **else** return 0;
- 34     Add block to the blockchain;
- 35     Share the computed HASH with all miners;
- 36 **Miner\_competitive(block);**
- 37 Miners compete to mine block;
- 38 **if** block is mined **then**
- 39     Add block to the blockchain;
- 40     Share the computed HASH with all miners;
- 41     **else** return 0;
- 42 **Threshold\_Update**(Payload size,
- 43  $MiningTime_{mean}, \sigma$ );

---

TABLE I: Mining type decision by the scheduler

$\mu^{TTL}$	$\mu^P$	$\mu^e$	Decision
Low	-any-	-any-	Collaborative
Medium	Low	Low	Competitive
High	Low	Low	Competitive
High	Low	Medium	Competitive
High	Medium	Low	Competitive
-rest-	-rest-	-rest-	Collaborative

TABLE II: Decision of mining to be collaborative or competitive, based on TTL, payload, and events

Events	$\mu_{T_m}(s)$	$Th(s)$	Valid	Expired	$\mu^{TTL}$	$\mu^P$	$\mu^e$	Type
202	0.0136	1.0311	164	38	M	L	M	Col.
43	0.1133	0.7023	39	4	M	L	L	Com.
266	0.0305	0.1031	262	4	M	L	H	Col.
65	0.3251	1.2839	54	11	M	L	L	Com.
264	0.0670	0.1813	254	10	M	L	H	Col.

mining strategy is favored in order to conserve resources and offer an equitable opportunity to all miners. Table I presents the decision mechanism of the mining type to be used. The events are inserted in Block ( $b_i$ ) having aggregated events  $\Lambda_{\tau\xi}$ , Block number ( $b_{i_n}$ ), time stamp, previous HASH. The process is presented in Algorithm 1, line 10 - 21.

3) *Miner Nodes:* The Block along with the mining decision is communicated to  $m_i$  in  $M_j$ . The miners start mining the block by finding the *nonce* according to the PoW for which a difficulty level ( $d$ ) is pre-defined.

**Competitive mining:** In the competitive mining approach, all miners in the clusters compete to mine the block by iterating the *nonce* to satisfy the PoW.

**Collaborative mining:** All the miners in clusters are assigned the upper bound ( $\chi_U$ ) and lower bound ( $\chi_L$ ) of nonce values which the miners use to mine the block. These miners only use the *nonce* values to mine the block within the defined range. The ranges are replicated among the miners as well to avoid any single point of failure for validating  $b_i$ .

A notification is transmitted in the clusters by the miner which successfully mines the block. Furthermore, the HASH is also communicated to all the clusters which is used for mining the next block. The HASH is also added to the ledger which is used for verification of the secure IIoT operations and  $b_i$  is added to the blockchain. This is presented in Algorithm 1, from line 23 to 37. The threshold is further updated with the current  $P$ ,  $\mu_{T_m}$  and  $\sigma_{T_m}$  as presented in line 38 in Algorithm 1.

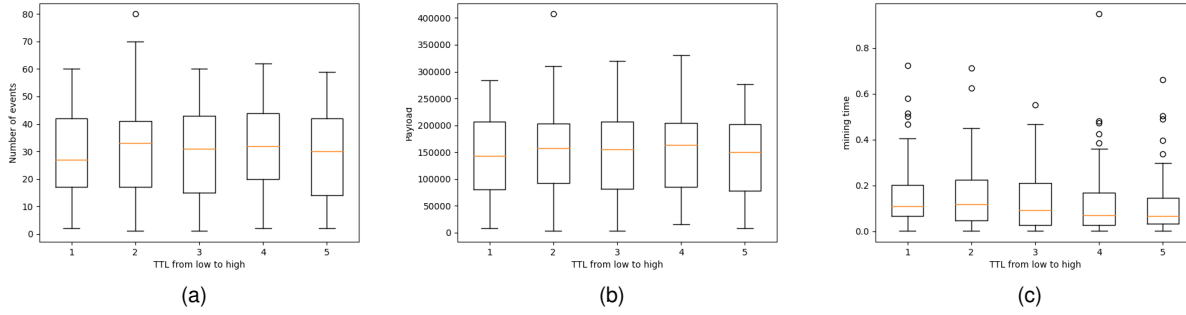


Fig. 2: Effect of TTL on  $e$ ,  $P$  and  $T_m$

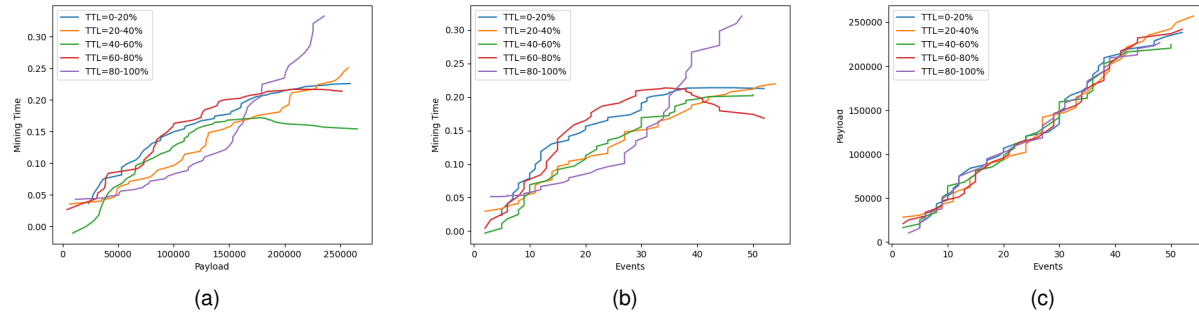


Fig. 3: Trends of  $P$ ,  $e$  and  $T_m$  for multiple TTL distributions

#### IV. RESULTS

Events are generated randomly having varying payload sizes, number of events, and TTL. These parameters are initially configured for simulation purposes, with specific motivations: (i) The selection of 600 events as the upper limit stems from the desire to assess the system's performance under substantial complexity and load, enabling the simulation of scenarios with a high event volume. (ii) A maximum payload size of 10KB is chosen to represent a realistic range of event sizes, aligning with the system's expected handling capacity. (iii) The TTL parameter's choice is grounded in the anticipated system behavior, facilitating the examination of the system's responsiveness to events with varying lifespans, particularly in time-sensitive scenarios. Table II presents the events generated for the first five instances. In the first instance, 202 events are generated by the transaction node. The  $\mu_{T_m}$  and  $Th$  were initially forecasted which are 0.0136 sec and 1.0311 sec. Here, 164 events are above the  $Th$  and are considered as valid to be added to the blocks while 38 events are below the  $Th$  due to which they are added to the list of expired events. Based on the average TTL, payload, and events, the scheduler node communicates to all the

clusters that the collaborative mining approach is used for the current batch of events. All the events are sorted per their TTL values and are bundled up in groups which are sent in ascending order to all the clusters for mining. Furthermore, the scheduler also computes the  $\mu_{T_m}$  and  $Th$  based on the current events. These will be used for the next batch of events as shown in Table II where 0.1133 and 0.7023 are used from the previous events added to the blockchain. It is observed that the  $\mu_{T_m}$  and  $Th$  decreases with less number of events and vice versa. As the distribution of events are random with having randomly associated payload sizes, the TTL does not show any direct relationship to these parameters such as  $P$  and  $e$  as shown in Figures 2a, and 2b. For the batches of events having ascending TTL, mining time is more uptill 60% of the TTL as shown in Figure 2c. The reason is that most of the events lie in that region having large payloads. The waiting time to mine the block will be large for larger TTL because the priority is assigned to mine the block with events having low TTL. To separately see the relationship of  $P$  and  $e$  with  $T_m$ , it is observed that by increasing  $P$  from 0KB to 250KB, and  $e$  from 0 to 50 for each TTL level, the  $T_m$  also increases as shown in Figures 3a and 3b. Miners

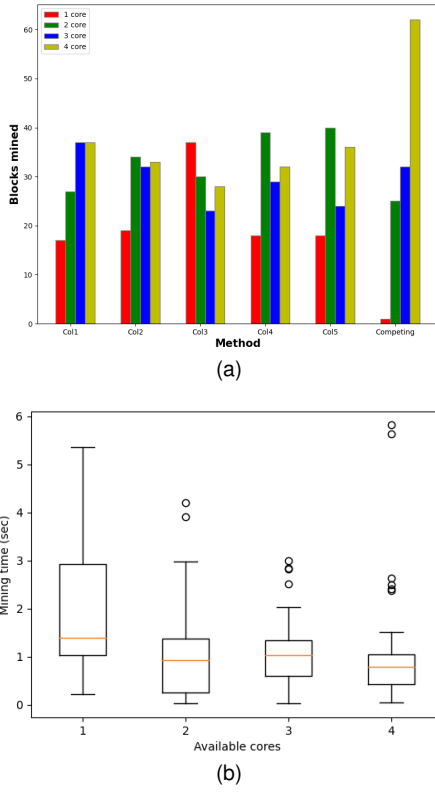


Fig. 4: Left: Distribution of blocks mined using different computational power; Right: Effect of computational power on  $T_m$  for  $D = 5$  and blocks = 40

require more time to find the right *nonce* which satisfies the PoW. Similarly, as the  $e$  increases, the payloads add up and reaches as high as 25KB when the event size reaches 50. This is presented in Figure 3c.

#### A. Competitive and Collaborative Mining

The comparison of competitive and collaborative mining methods is presented in Figures 4 and 5. In competitive mining, all miners in the clusters compete to mine the block while in collaborative mining, a pre-defined range of *nonce* values are used by the miners. Figure 4a presents the results generated for mining the blocks using both collaborative and competitive mining approaches. Miners have different computational capabilities in which they can use up to 4 cores to maximize their computational power. 120 blocks are mined by miners for both cases. In competitive mining, the miners start iterating the *nonce* from 0 till the required *nonce* value to verify the block for the defined PoW. The miners with high computational power reach the required *nonce* value quickly as compared to other miners with low computational capabilities. The rightmost bars in Figure 4a show the mined blocks by miners using 1

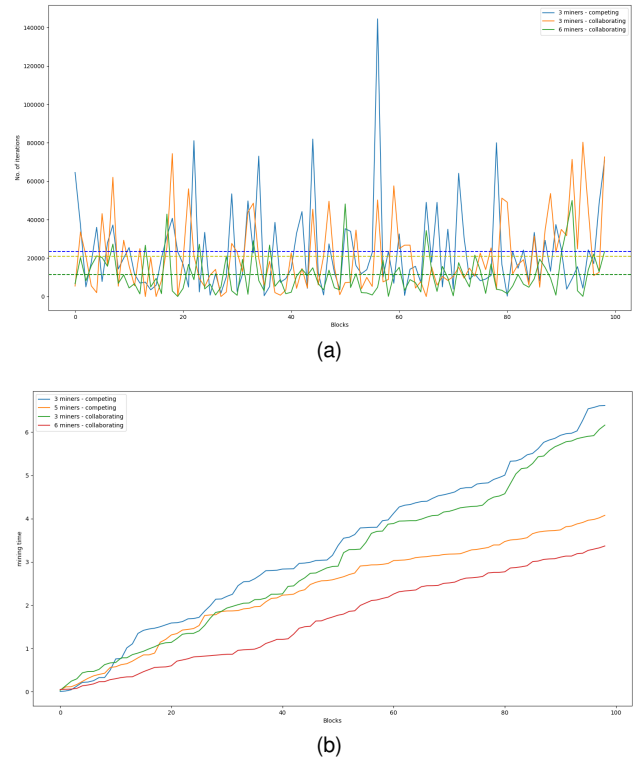


Fig. 5: Left: Blocks vs. number of nonce iterations; Right: Blocks vs. mining time

to 4 cores. 61 blocks are mined by miners with maximum computational capabilities whereas, only 1 block is mined by miners with fewer resources. As the *nonce* ranges are already assigned to the miners in collaborative mining, the computational capabilities of the miners have no obvious effect on the mining time of the blocks. It depends on how far the value of *nonce* is from the  $\chi_L$  as it can have multiple values to solve the PoW puzzle. Due to this reason, a random trend is achieved each time which does not depend on computational power but rather the distance of *nonce* from the  $\chi_L$ . For a single miner to solve the PoW puzzle, the mining time decreases as the computational power increases. This is presented in 4b in which same 40 blocks are assigned to four miners having different computational capabilities. Miner using a single core takes a  $\mu_{T_m}$  and maximum mining time ( $T_{m(max)}$ ) of 1.3 sec and 5.3 sec, respectively, whereas, a miner with maximum potential (using 4 cores) has a  $\mu_{T_m}$  and  $T_{m(max)}$  as 0.9 sec and 1.5 sec, respectively. Compared to the state-of-the-art presented in [16] where the mining time is achieved as  $\approx 0.9$  sec to mine a single block keeping  $d = 4$ , in our model it is achieved as low as  $\approx 0.1$  sec as presented in Figure 4b. Furthermore, in [16], the payload size is fixed, whereas in our model, the realistic range of variable payload is considered. Figure 5 presents the comparison



of competing and collaborative mining for the mining time by increasing the number of miners. In Figure 5a, the mean *nonce* iterations are presented for 3 miners and 6 miners. It is observed that the iterations of *nonce* are more for competitive mining compared to collaborative for the same number of miners because the iterations start from 0 in competitive mining. The results shown are for  $d = 4$ . For a higher value of  $d$ , the gap between a mean number of iterations of both mining types will be more because the value of *nonce* is usually high for higher  $d$ . In Figure 5b, as the number of blocks increases, the mining time also increases. More the number of iterations, the more the mining time, hence it is clear that for the same number of miners, competitive mining requires more mining time compared to collaborative because the *nonce* iterations are less for the latter.

## V. CONCLUSIONS

This paper has introduced the idea of enhancing IIoT events' logging security using blockchain. The proposed work has integrated sensor events into blockchain clusters, combated outdated risks with a dynamic threshold, and favored collaborative mining for low TTL scenarios. Overall, the proposed work has improved security and IIoT data management. Looking ahead, our future research directions involve optimizing the network infrastructure and exploring different consensus algorithms. Additionally, we plan to integrate the current model with edge computing to reduce latency and implement advanced threat detection mechanisms, which are expected to further enhance system performance and reduce security risks.

## REFERENCES

- [1] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "Guide to Industrial Control Systems (ICS) Security, 2015," *NIST Special Publication*, pp. 800–82, 2023.
- [2] M. Landauer, F. Skopik, M. Frank, W. Hotwagner, M. Wurzenberger, and A. Rauber, "Maintainable Log Datasets for Evaluation of Intrusion Detection Systems," *IEEE Transactions on Dependable and Secure Computing*, 2022.
- [3] D. U. Case, "Analysis of the Cyber Attack on the Ukrainian Power Grid," *Electricity Information Sharing and Analysis Center (E-ISAC)*, 2016.
- [4] D. Ma and G. Tsudik, "Forward-Secure Sequential Aggregate Authentication," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 86–91.
- [5] J. E. Holt, "Logcrypt: Forward Security and Public Verification for Secure Audit Logs," in *Proceedings of the 2006 Australasian workshops on Grid computing and e-research-Volume 54*. Australian Computer Society, Inc., 2006, pp. 203–211.
- [6] A. Meryem and B. E. Ouahidi, "Hybrid Intrusion Detection System using Machine Learning," *Network Security*, vol. 2020, no. 5, pp. 8–19, 2020.
- [7] M. Castro, B. Liskov *et al.*, "Practical Byzantine Fault Tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186.
- [8] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A Survey of Distributed Consensus Protocols for Blockchain Networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.
- [9] I. Malakhov, A. Marin, S. Rossi, and D. Smuseva, "On the use of Proof-of-Work in Permissioned Blockchains: Security and Fairness," *IEEE Access*, vol. 10, pp. 1305–1316, 2021.
- [10] A. F. Anta, K. Konwar, C. Georgiou, and N. Nicolaou, "Formalizing and Implementing Distributed Ledger Objects," *ACM SIGACT News*, vol. 49, no. 2, pp. 58–76, 2018.
- [11] M. Kamal, G. Srivastava, and M. Tariq, "Blockchain-Based Lightweight and Secured V2V Communication in the Internet of Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 3997–4004, 2020.
- [12] D. L. Mills, "Internet Time Synchronization: The Network Time Protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, 1991.
- [13] J. Eidson and K. Lee, "IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," in *Sensors for Industry Conference, 2002. 2nd ISA/IEEE*. IEEE, 2002, pp. 98–105.
- [14] H. Heitsch and W. Römsch, "Scenario Tree Modeling for Multistage Stochastic Programs," *Mathematical Programming*, vol. 118, no. 2, pp. 371–406, 2009.
- [15] Y. Liu, K. Wang, Y. Lin, and W. Xu, "LightChain: a Lightweight Blockchain System for Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3571–3581, 2019.
- [16] V. Mardiansyah and R. F. Sari, "SimBlock Simulator Enhancement with Difficulty Level Algorithm Based on Proof-of-Work Consensus for Lightweight Blockchain," *Sensors*, vol. 22, no. 23, p. 9057, 2022.



**Mohsin Kamal (M'16–SM'20)** is designated as Associate Professor in School of Electrical Engineering and Computer Science (SEECs), National University of Science and Technology (NUST). His research interests include the development of lightweight solutions for various IIoT applications, Cyber Physical Systems, Blockchain, Cyber Security, Wireless Sensor Networks, Cooperative Communication and Cognitive Radio Networks.



**Muhammad Tariq (M'12–SM'17)** holds the position of Professor and Head at the Department of Electrical Engineering, NUCES, Islamabad, Pakistan. His research portfolio includes publications in the Internet of things, network sciences, cybersecurity, and smart grids, published in prestigious IEEE journals/transactions. He earned recognition as being among Stanford's university list of the World's Top 2% Scientists.



**Mian Ahmad Jan (M'16–SM'20)** is an Assistant Professor at the Department of Computer Science, School of Computing and Informatics, University of Sharjah, United Arab Emirates. His research interests include cybersecurity, energy-efficient and secured communication for Wireless Sensor Networks, and the Internet of Things. He has been among Stanford's university list of the World's Top 2% Scientists.



**Houbing Herbert Song (M'12–SM'14–F'23)** is a Professor and the Director of the NSF Center for Aviation Big Data Analytics (Planning), and the Director of the Security and Optimization for Networked Globe Laboratory, University of Maryland, Baltimore, MD. His research interests include cyber-physical systems/internet of things, cybersecurity and privacy, and AI/machine learning/big data analytics.