EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

# A Private-Blockchain and Edge–Fog Authentication Framework for Coal Mine Safety Terminals

Mei Zhang[1], Fangting Zheng[1,*], Jianshu Hu[1]

[1]School of Electrical and Information Engineering, Anhui University of Science and Technology, Huainan 232001, China

## Abstract

Coal-mine safety terminal networks require trustworthy and scalable access in heterogeneous underground settings. This paper presents a private-blockchain and edge–fog authentication framework combining static–dynamic terminal identifiers, a three-network-message SM4-GCM/HMAC authentication protocol, and locally processed authentication with ledger recording outside the immediate processing path. PoS-weighted committee selection, fog failover, permission smart contracts, and cross-chain audit are specified as architectural designs, rather than implemented subsystems. A single-host multiprocessing prototype was evaluated with batches of 100–2000 submitted requests. At 2000 requests, its observed mean request turnaround time was 125.85 ms and throughput was 8608.77 requests/s, versus 128.94 ms and 8185.56 requests/s for a simplified PBFT-style comparator; Welch tests across five runs found neither difference statistically significant ($p=0.6761$ and 0.4383). A separate reproducible controlled synthetic anomaly study yielded CNN–LSTM accuracy of $0.910\pm0.005$ and Macro-F1 of $0.906\pm0.009$ over three seeds. These findings characterize a software prototype and simulated data only, not a deployed mine authentication system or end-to-end underground link performance.

**Keywords:** scalable authentication, private blockchain, edge-fog computing, Industrial Internet of Things, coal mine safety terminal, hybrid identifier

Received on 07 August 2026, accepted on 03 November 2026, published on 27 November 2026

Copyright © 2026 Mei Zhang *et al.*, licensed to EAI. This is an open access article distributed under the terms of the CC BY-NC-SA 4.0, which permits copying, redistributing, remixing, transformation, and building upon the material in any medium so long as the original work is properly cited.

doi: 10.4108/eetsis.16886

## 1. Introduction

With the continued development of intelligent coal mining, gas sensors, personnel locators, video-surveillance devices, equipment controllers, and emergency-communication terminals have been widely deployed underground, forming a pervasive sensing and control network that covers personnel, equipment, and the operating environment [1]. Such coal-mine safety terminal networks can be regarded as a representative distributed Industrial Internet of Things (IIoT) information system, in which heterogeneous devices are deployed across multiple underground regions and continuously exchange safety-critical data and control information [2,3]. As the number of terminals and concurrent access requests increases, the authentication subsystem must maintain trustworthy access, low response latency, and service continuity under growing system scale [4,5]. Safety terminals are not only sources of production and environmental data but also actuators of control commands; their identity legitimacy therefore directly affects the authenticity of safety-monitoring data and the reliability of on-site control.

These characteristics create challenges for scalable authentication in underground environments. Some centralized management and key-generation approaches expose an availability or trust bottleneck, as discussed in

*Corresponding author. Email: 1297740238@qq.com

IIoT authentication research [5,6]; the prevalence of any specific coal-mine architecture requires deployment-level evidence. In addition, hardware codes, device numbers, and network addresses assigned by different vendors may lack unified mapping, complicating cross-system recognition. Consequently, an authentication framework should address identity legitimacy, access continuity, heterogeneous context, and verifiable audit while distinguishing system design from demonstrated implementation.

Blockchain provides distributed storage, tamper resistance, and traceability, whereas edge-fog computing moves authentication and decision-making closer to underground operating areas [7,8]. Their integration can reduce centralized dependence and cloud round trips while providing a trustworthy data foundation for terminal-state management and accident traceability [9,10].

Representative IIoT authentication approaches address complementary aspects of this problem. Blockchain-based certificateless cross-domain authentication has been studied to reduce dependence on centralized key-generation authorities [6]; edge-assisted authentication uses nearby computation for resource-constrained devices [8], while a decentralized smart-factory architecture uses blockchain to distribute inter-domain trust [5]. Coal-mine hazard monitoring studies [2,3,11] establish the sensing context but should not be assumed to validate the same authentication protocol. The present contribution is a design-level combination of context-sensitive identifiers, a three-message local authentication exchange and auditable state management. Only the local authentication software prototype and a controlled synthetic anomaly classifier are quantitatively exercised here; full consensus, contracts and cross-chain deployment remain proposed extensions.

This paper proposes a private-blockchain and edge–fog authentication framework for coal mine safety terminals. Its design combines static terminal attributes with dynamic environmental context, specifies PoS-weighted committee selection with PBFT-based registration, and places routine mutual authentication at authorized fog nodes. Dynamic permission contracts, CNN–LSTM monitoring, fog-node failover and Merkle-based cross-chain traceability are architectural components whose full implementations remain future work. The three-message protocol is analyzed under stated BAN-logic assumptions. For evaluation, a controlled single-host prototype measures centralized, local-fog and simplified PBFT-style processing under fixed submitted-request batches; a separately generated synthetic dataset supports a reproducible CNN/LSTM/CNN–LSTM comparison. These measurements cannot establish mine-field latency or a performance advantage that is statistically significant under the five-run sample.

The remainder of this paper is organized as follows. Section 2 presents the system architecture, design objectives, and registration and authentication workflow. Section 3 describes hybrid-identifier generation, hierarchical consensus, fog-node collaboration, dynamic permission control, CNN-LSTM-based anomaly detection, and cross-chain traceability. Section 4 analyzes the three-message authentication protocol using BAN logic and complementary attack-resistance analysis. Section 5 evaluates the proposed framework through functional comparison, protocol-efficiency analysis, scalability and ablation experiments, and anomaly-detection evaluation. Finally, Section 6 summarizes the main findings and discusses remaining limitations and future work.

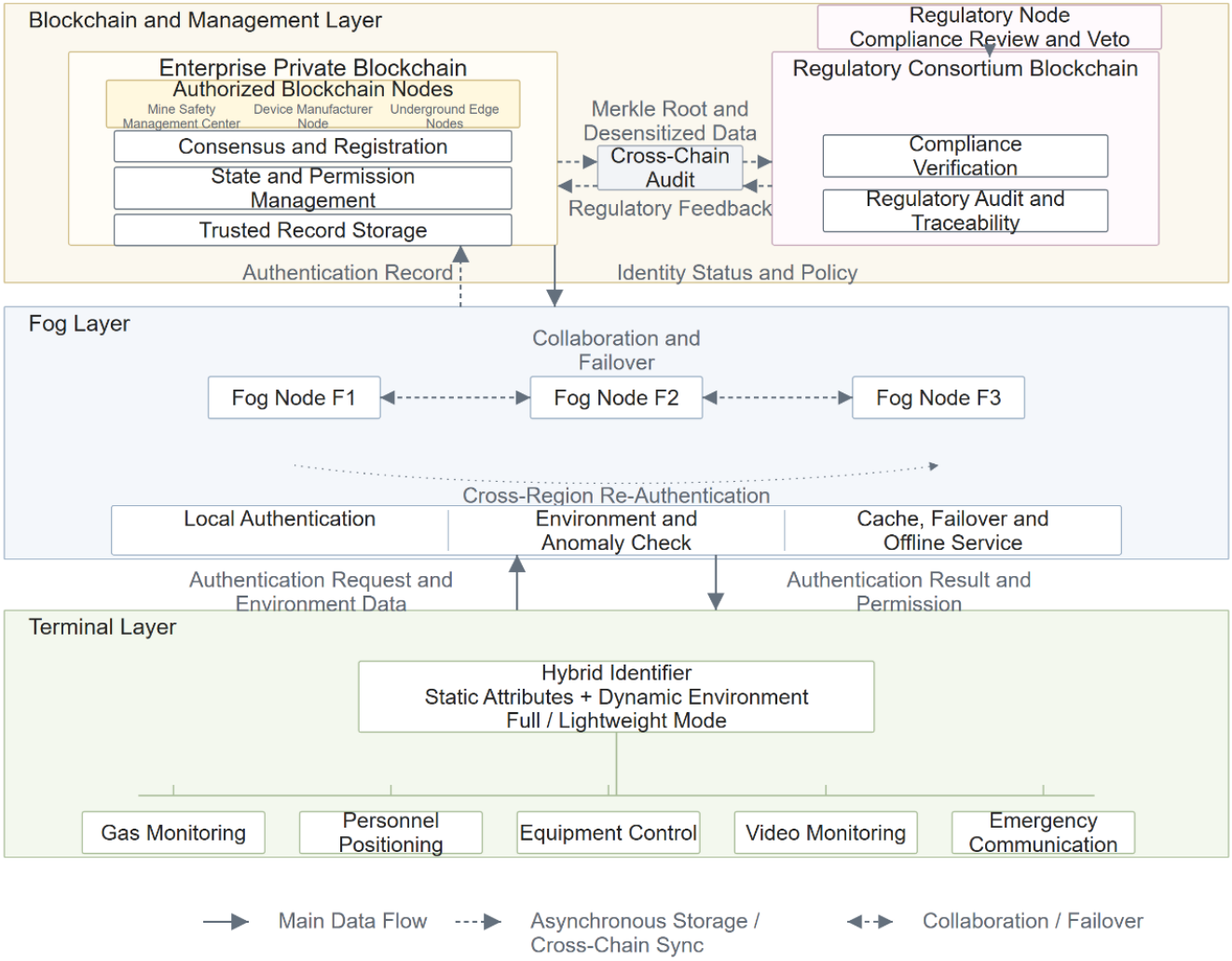


**Figure 1.** Overall architecture of the coal mine safety terminal authentication system

## 2. System architecture

This section introduces the three-layer system architecture, security assumptions, and the registration and authentication procedure that support the mechanisms developed in the following sections.

### 2.1 Three-layer architecture

The proposed system adopts a terminal–fog–cloud/blockchain architecture. The terminal layer collects terminal and environmental information, the fog layer provides near-end authentication and failover, and the cloud/blockchain layer supports trusted registration, state management, and traceability. The overall architecture and principal data exchanges are shown in Figure 1.

As shown in Figure 1, the system comprises the terminal layer, fog layer, and blockchain-and-management layer. The terminal layer collects static terminal attributes, operating states, and dynamic environmental information and submits authentication requests to fog nodes. The fog layer performs local authentication, environmental-consistency verification, cache reuse, cross-region re-authentication, and failover. The blockchain-and-management layer maintains registration consensus, terminal states, permission policies, audit records, and traceability. Solid arrows represent the main authentication data flow, dashed arrows indicate asynchronous recording or cross-chain synchronization, and bidirectional dashed arrows indicate fog-node collaboration and failover.

### 2.2 Design objectives and security assumptions

The three-layer architecture defines the responsibilities and information exchanges of the participating entities, but secure operation also depends on explicit authentication objectives, adversarial capabilities, and trust boundaries.

Accordingly, the system is designed to provide trusted identity, secure communication, service continuity, and behavioral traceability. A terminal and a fog node should mutually authenticate each other and establish a fresh key bound to the current session. Nonces and timestamps are used to resist replay attacks. The failure of a single center or fog node should not cause a system-wide outage, and registration, authentication, permission adjustment, and anomaly-handling events should all produce tamper-evident records.

The adversary is assumed to be capable of eavesdropping on, modifying, blocking, and replaying open-channel messages between terminals and fog nodes, and may control some consensus nodes or attempt to register forged terminals. Static-attribute protection keys, terminal-domain master keys, fog-node-specific authentication keys, and the direction-specific message-protection keys derived from them are stored or transmitted through protected cryptographic modules and secure channels and cannot be obtained by the adversary within a feasible time. The number of Byzantine nodes in a private-chain committee does not exceed the PBFT fault-tolerance bound. Regulatory nodes and the mine safety management center are regarded as trusted during registration review and compliance arbitration.

## 2.3 Registration and authentication procedure

Under the preceding objectives and assumptions, the proposed access procedure has three stages: registration recording, long-term-key configuration, and online authentication. The registration design sends terminal qualifications and environmental-reference metadata through a fog node to a private ledger. A PoS-weighted 21-member committee and PBFT verification are specified for eventual deployment, but the experiments in Section 5 do not implement this committee. The on-chain reference tuple points to a separately controlled off-chain environmental vector and is defined as

$$Ref_T^b = (Ptr_T^b, \mathrm{Ver}_T^b, \mathrm{H}[\mathrm{Encode}(x_T^b, t_b)]) \quad (1)$$

In Eq. (1), $Ref_T^b$ denotes the most recent trusted environmental-reference tuple for terminal $T$; the superscript $b$ identifies the baseline trusted state; $Ptr_T^b$ is the access pointer to the off-chain environmental data; $Ver_T^b$ is the data version; $x_T^b$ is the quantized trusted environmental vector; $t_b$ is its sampling time; Encode $(\cdot)$ denotes deterministic encoding; and H $(\cdot)$ denotes a cryptographic hash function. The vector is stored in a controlled off-chain environmental database or a trusted fog cache, whereas only its pointer, version, and digest are recorded on-chain.

Following registration, the management-center design provisions a distinct terminal-specific master secret K_TD for each terminal T. K_TD is never shared among terminals. A terminal–fog long-term key is then derived using the terminal base identifier, authorized fog identifier and key version:

$$K_{TF} = \mathrm{KDF}\left(K_{TD}, ID_T^{\mathrm{base}} \parallel ID_F \parallel v_K\right) \quad (2)$$

In Eq. (2), $K_{TD}$ is the terminal-domain master key; $K_{TF}$ is the long-term authentication key shared by terminal $T$ and authorized fog node $F$; $ID_T^{\mathrm{base}}$ is the stable base identifier; $ID_F$ is the fog-node identifier; $v_K$ is the key version; KDF $(\cdot)$ denotes the key-derivation function; and "‖" denotes field concatenation. Binding the terminal identity, fog-node identity, and key version to the derivation process separates keys across terminals, fog nodes, and key versions [12].

The terminal stores $K_{TD}$ in its secure module and can independently derive $K_{TF}$. The mine safety management center securely delivers the corresponding $K_{TF}$ to a fog node that has obtained on-chain authorization. Neither the domain master key nor the long-term authentication key is recorded directly on-chain; only the key index, version, and validity-state digest are stored. The static-attribute protection key $K_T$ and the long-term authentication key $K_{TF}$ serve different purposes and must not be reused.

In the proposed key-isolation policy, compromise of the secure storage of one terminal exposes that terminal's K_TD but does not reveal any other terminal's independently provisioned K_TD. The management center can revoke the affected key index, increment its version and securely re-provision that terminal and authorized fog nodes. Cross-region handover delivers only the target-fog-specific derived key over a protected management channel. These are design and trust assumptions; compromise resilience of secure elements and revocation propagation have not been experimentally evaluated.

After registration recording and long-term-key configuration, the terminal enters the online authentication stage. To provide mutual authentication and explicit key confirmation with a limited number of communication rounds, local key preparation precedes three network messages: terminal request M1, fog response M2, and terminal confirmation M3. The terminal derives the long-term authentication key and two direction-separated message-protection keys, generates a fresh nonce and timestamp, and encapsulates the current hybrid identifier, complete environmental message, and freshness parameters in the first request message $M_1$. After receiving $M_1$, the fog node locates the latest trusted off-chain environmental vector through the on-chain reference tuple, verifies its digest, and performs environmental-consistency checking using neighboring terminals that are in the normal state and within the same sampling window. If verification succeeds, the fog node generates its own fresh nonce and returns the terminal state, session parameters, and fog-node key-confirmation value in $M_2$. The terminal then sends $M_3$ as explicit key confirmation. After $M_3$ is verified, the session is established and the authentication result is recorded asynchronously. The direction keys, message fields, and confirmation values are specified in Section 4.2.

The registration and three-message authentication workflows are illustrated in Figure 2.

To further clarify the complete interaction from initial registration to session establishment, Figure 2 presents the sequence of the terminal-registration stage and the three-message authentication stage.

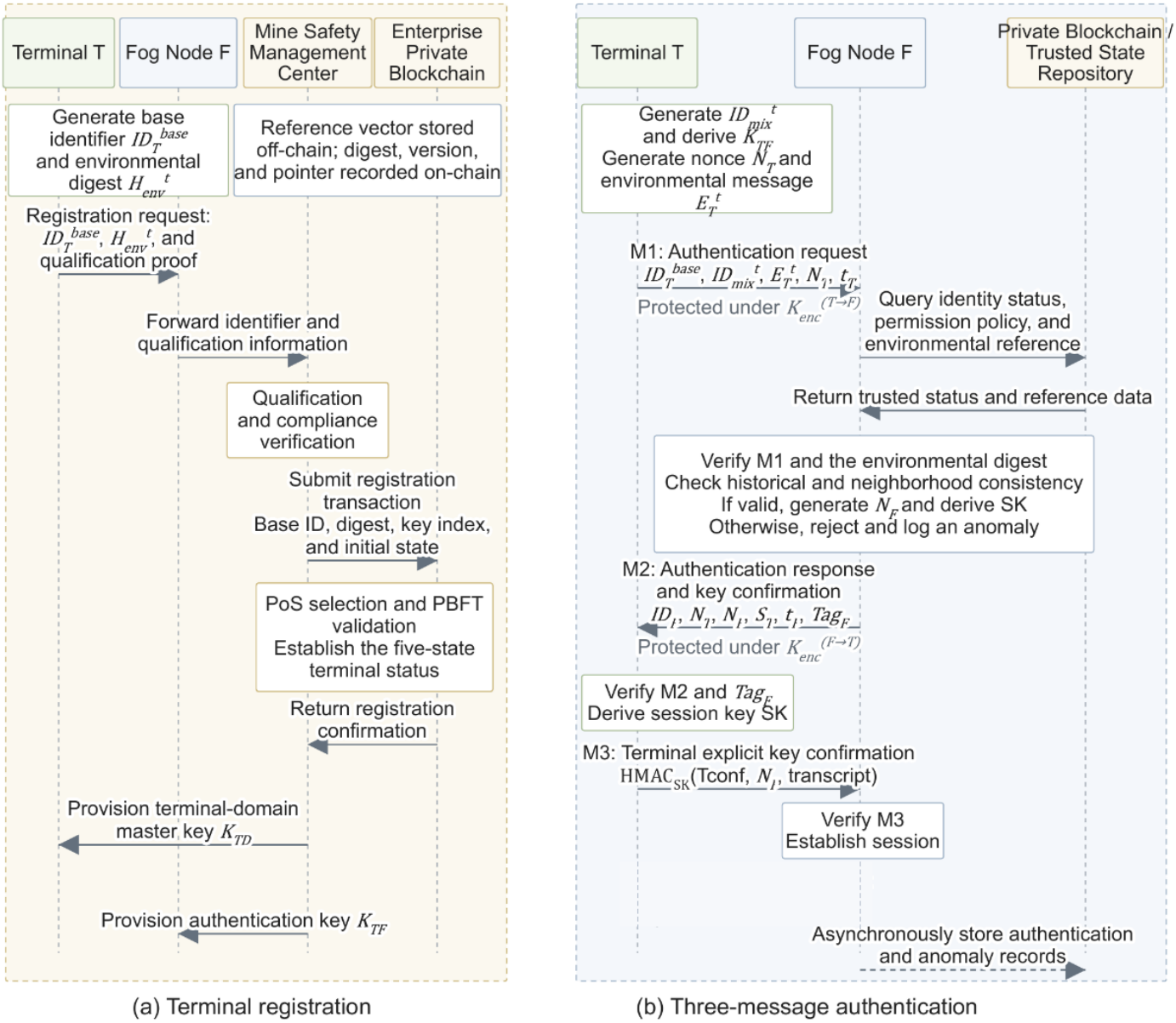


**Figure 2.** Terminal registration and three-message authentication

Figure 2(a) shows terminal registration. The terminal first generates its stable base identifier and current environmental digest, which are forwarded through a fog node to the enterprise private blockchain for qualification review and PoS/PBFT verification. The environmental reference vector is stored in a controlled off-chain repository, while only its digest, version, and access pointer are recorded on-chain. After registration, the mine safety management center provisions the terminal-domain master key to the terminal and securely distributes the dedicated long-term authentication key to the authorized fog node. Figure 2(b) shows the three-message authentication procedure: the terminal uses $M_1$ to submit the current hybrid identifier, environmental message, and freshness parameters; the fog node returns the authentication response and its key-confirmation value in $M_2$ after checking the identity state, permission policy, and environmental consistency; and the terminal sends $M_3$ for

explicit key confirmation. After session establishment, authentication results and anomaly records are asynchronously recorded on-chain.

In Figure 2(b), querying the trusted repository and asynchronously submitting an audit record are fog-side back-end actions, not additional terminal–fog authentication messages; the terminal-facing protocol contains only M1, M2, and M3. Permission decisions are local outputs after M3 is verified, not a fourth authentication response.

## 3. System design

Section 2 defined the overall architecture and authentication interaction. This section explains how the required functions are implemented. Hybrid identifiers and environmental-consistency verification first establish a trustworthy representation of terminal identity and state, after which hierarchical consensus records registration results. Fog nodes then provide near-end authentication and failover; smart contracts adjust permissions according to terminal, environmental, and operator conditions, and anomaly detection together with cross-chain auditing supports continuous monitoring and post-event traceability. These mechanisms form an operational closed loop from identity generation and authentication to authorization, response, and audit.

### 3.1 Hybrid identifier generation

The hybrid identifier adopts a two-layer structure consisting of a static base layer and a dynamic environmental layer. The static layer retains inherent hardware features, such as the chip unique identifier $UID_T$ and core-module serial number $SN_T$; incorporates contextual attributes including deployment area $Area_T$, device type $\mathrm{Type}_T$, and security level $L_T$; and links them to the private-chain address $Addr_T$, initial access timestamp $t_0$, and security-qualification number $Cert_T$. To avoid ambiguity caused by direct field concatenation, all fields are first deterministically encoded according to a fixed order, data type, and length:

$$M_T = \mathrm{Encode}(UID_T, SN_T, Area_T, \mathrm{Type}_T, L_T, Addr_T, t_0, Cert_T) \quad (3)$$

$$(C_T, Tag_T) = \mathrm{SM4-GCM.Enc}_{K_T}(IV_T, M_T, AD_T) \quad (4)$$

In Eqs. (3) and (4), $M_T$ denotes the deterministically encoded static-attribute message of terminal $T$; $UID_T$ is the chip unique identifier; $SN_T$ is the core-module serial number; $Area_T$, $Type_T$, and $L_T$ denote the deployment area, terminal type, and security level; $Addr_T$ is the terminal address on the enterprise private blockchain; $t_0$ is the first-access timestamp; and $Cert_T$ is the security-qualification number. $C_T$ and $Tag_T$ are the ciphertext and authentication tag generated by SM4-GCM; $K_T$ is the static-attribute protection key; $IV_T$ is an initialization vector that must be unique under $K_T$; and $AD_T$ is associated data that remain in plaintext but are integrity protected. $K_T$ is used only for static-attribute protection and must not be reused as $K_{TF}$.

Eq. (3) defines deterministic encoding of the static attributes. Eq. (4) applies GCM-based authenticated encryption over the SM4 block cipher to provide confidentiality and integrity [13]. The key $K_T$ is used only to protect static attributes. The initialization vector must be unique under the same key, and the associated data protects fields that need not be encrypted but must be authenticated.

The GCM mode follows NIST SP 800-38D [14], and the SM4 primitive follows GB/T 32907-2016 [15].

The dynamic environmental layer encodes gas concentration, humidity, geological stress, location, and their respective sampling timestamps using unified units, precision, and field lengths to obtain the environmental message $E_T^t$. A dynamic environmental digest is then computed according to SHA-256:

SHA-256 follows the Secure Hash Standard [16].

$$E_T^t = \mathrm{Encode}(G_t, t_G, \mathrm{Hum}_t, t_{\mathrm{Hum}}, P_t, t_P, Loc_t) \quad (5)$$

$$\mathrm{H}_{\mathrm{env}}^t = \mathrm{SHA-256}(E_T^t) \quad (6)$$

In Eqs. (5) and (6), $E_T{}^t$ denotes the complete environmental message generated by terminal $T$ at time $t$; $G_t$ and $t_G$ are the gas concentration and its sampling time; $Hum_t$ and $t_{Hum}$ are the humidity and its sampling time; $P_t$ and $t_P$ are the geological stress and its sampling time; and $Loc_t$ is the terminal location. $H_{\mathrm{env}}{}^t$ is the SHA-256 digest of $E_T{}^t$. The digest verifies message integrity and is not used directly in numerical-deviation calculations.

To ensure that historical records for the same terminal remain retrievable after environmental changes, the on-chain base identifier $ID_T^{\mathrm{base}}$ is generated from the static ciphertext, authentication tag, initialization vector, associated-data digest, blockchain address, first-access time, and qualification verification code. It is then combined with the identifier mode and dynamic environmental digest to form the current hybrid identifier $ID_{\mathrm{mix}}^t$:

$$ID_T^{\mathrm{base}} = H(\mathrm{Encode}(C_T, Tag_T, IV_T, H(AD_T), Addr_T, t_0, CV_T)). \quad (7)$$

$$ID_{\mathrm{mix}}^t = \mathrm{Encode}(ID_T^{\mathrm{base}}, Mode_T, \mathrm{H}_{\mathrm{env}}^t) \quad (8)$$

In Eqs. (7) and (8), $ID_T{}^{\mathrm{base}}$ denotes the stable base identifier; $C_T$, $Tag_T$, and $IV_T$ are the static-attribute ciphertext, authentication tag, and initialization vector; $H(AD_T)$ is the digest of the associated data; $CV_T$ is the qualification-verification code; $Mode_T$ distinguishes the full and lightweight identifier modes; and $ID_{\mathrm{mix}}{}^t$ is the current hybrid identifier. Environmental or location changes update $H_{\mathrm{env}}{}^t$ and $ID_{\mathrm{mix}}{}^t$, while $ID_T{}^{\mathrm{base}}$ remains unchanged to preserve historical-query continuity.

The identifier mode distinguishes the full and lightweight modes. $ID_T^{\text{base}}$ serves as a stable on-chain query index and binds the initialization vector, authentication tag, and associated-data digest used for static-attribute protection to the same identity structure, whereas $H_{\text{env}}^t$ is updated when the environment or location changes. To improve robustness against strong electromagnetic interference, the hybrid identifier uses repetition coding combined with interleaving: Repetition and interleaving are optional channel-protection design elements; no interleaver depth or coding gain was implemented or calibrated in the present software tests. Core control terminals use the complete field set, whereas ordinary sensing terminals retain only the core fields.

The dynamic environmental digest is used to construct the current hybrid identifier and to determine whether an environmental message has been altered during transmission or storage. Digest equality, however, establishes only message integrity and cannot quantify the deviation of current environmental values from a trusted historical state. The system therefore combines the controlled off-chain historical vector with data from trusted neighboring terminals to perform numerical-consistency verification. Let the current environmental vector be x_T^t. The latest trusted vector x_T^b is located through the on-chain reference tuple and can serve as a historical reference only after its recomputed digest matches the on-chain record. The historical-consistency distance, neighborhood-consistency distance, and combined validation condition are defined in Eqs. (9)–(11), respectively.

$$D_{\text{his}} = \sum_{k=1}^{m} \omega_k \frac{\left|x_{T,k}^t - x_{T,k}^b\right|}{x_k^{max} - x_k^{min} + \varepsilon} \tag{9}$$

$$D_{\text{nei}} = \sum_{k=1}^{m} \omega_k \frac{\left|x_{T,k}^t - \text{median}(x_{1,k}^t, x_{2,k}^t, x_{3,k}^t)\right|}{x_k^{max} - x_k^{min} + \varepsilon} \tag{10}$$

$$Valid_{\text{env}} = [\text{SHA} - 256(E_T^t) = H_{\text{env}}^t] \wedge (D_{\text{his}} \le \tau_1) \wedge (D_{\text{nei}} \le \tau_2) \tag{11}$$

In Eqs. (9) and (10), $D_{\text{his}}$ and $D_{\text{nei}}$ denote the historical- and neighborhood-consistency distances; $m$ is the number of environmental features; $k$ is the feature index; $\omega_k$ is the weight of the kth feature; $x_{T,k}^t$ and $x_{T,k}^b$ are the kth components of the current and trusted vectors; $x_k^{max}$ and $x_k^{min}$ are the corresponding range limits; ε is a small positive constant; and median(·) is the median operator. In Eq. (11), $Valid_{\text{env}}$ is the combined validation result, $\tau_1$ and $\tau_2$ are the historical- and neighborhood-consistency thresholds, and "∧" denotes logical conjunction.

Eq. (9) measures the range-normalized deviation between the current environment and the latest trusted state, whereas Eq. (10) uses the median of three trusted terminals in the same area as the neighborhood reference. A terminal included in the neighborhood comparison must satisfy $S_j = \text{normal}$ and $|t_j - t| \le \Delta t$, thereby ensuring a normal state and comparable sampling time. Eq. (11) recomputes the SHA-256 digest of $E_T^t$ and simultaneously evaluates $D_{\text{his}}$ and $D_{\text{nei}}$ . The thresholds should be calibrated using sensor ranges, measurement uncertainty, and representative underground operating data. The dimensionless thresholds τ1 and τ2 are independently configurable and have not been jointly calibrated for mine-field use; no universal 5% value is claimed.

## 3.2 Hierarchical consensus and registration recording

Hybrid identifiers and environmental-consistency verification describe the terminal identity and its current environmental state, but these results must still be confirmed by multiple parties and converted into tamper-evident on-chain records. Registration therefore combines PoS-based node selection with PBFT transaction verification. Regulatory nodes, mine-safety-management-center nodes, device-manufacturer nodes, and underground edge nodes are assigned initial weights of 30%, 25%, 20%, and 25%, respectively, while the weight of any individual edge node is limited to 10%. The probability that candidate node i is selected is

$$p_i = \frac{w_i}{\sum_{j=1}^{N_{\text{node}}} w_j} \tag{12}$$

In Eq. (12), $p_i$ is the probability that candidate node $i$ is selected for the consensus committee, $w_i$ is its initial weight, $N_{\text{node}}$ is the total number of candidate nodes, and $j$ is the summation index.

The system selects a 21-node committee according to these weights. PoS weights are used only for committee selection; once selected, nodes cast equal votes in the PBFT committee. The PBFT fault bound follows the standard model, while the confirmation threshold is configured to exceed two-thirds of the committee [17]. Recent studies have shown that hierarchical and grouping strategies can reduce PBFT communication overhead and improve scalability in IoT and large blockchain networks [18,19]. Accordingly, the fault-tolerance bound and confirmation threshold adopted in this system are defined as follows:

The committee's assumed Byzantine model follows the foundational PBFT literature [20].

$$f = \text{floor}\left(\frac{n-1}{3}\right) \tag{13}$$

$$n = 21 \quad \Rightarrow \quad f = \text{floor}\left(\frac{20}{3}\right) = 6 \tag{14}$$

$$q = n - f = 21 - 6 = 15 > \frac{2n}{3} \tag{15}$$

In Eqs. (13)-(15), $n$ is the PBFT committee size; $f$ is the maximum number of Byzantine nodes that can be tolerated; floor(·) denotes downward rounding; $q$ is the transaction-confirmation threshold; and "⇒" indicates the

result obtained after substituting the committee size used in the proposed configuration.

Accordingly, a 21-node committee can tolerate at most six Byzantine nodes. Standard PBFT generally requires no fewer than 2f+1 consistent nodes for safety; this system adopts a stricter 15-node confirmation threshold. The regulatory veto is a compliance-governance mechanism and may affect liveness when the regulatory node is unavailable. Zero-knowledge proofs are limited to Boolean predicates such as valid qualification and absence from a violation list, thereby avoiding disclosure of raw information [21]. Differential privacy is used only for aggregated regulatory statistics and reports and does not perturb the identity, state, thresholds, or real-time authentication result of an individual terminal [22,23]. The stable base identifier links the five-state lifecycle and violation records.

The 21-member committee, weighted selection, regulatory veto, zero-knowledge checks, and differential-privacy statistics in this subsection are design specifications only. The software comparator in Section 5 uses four local logical validators rather than the proposed registration consensus architecture.

To specify the initial weights and functional responsibilities of the node types involved in committee selection, Table 1 summarizes the configuration of regulatory, mine-safety-management, device-manufacturer, and underground-edge nodes.

Table 1. Proposed weight configuration for hierarchical consensus nodes

| Node type | Weight | Primary responsibilities |
|---|---|---|
| Regulatory node | 30% | Compliance verification, historical-violation checking, and consensus veto |
| Mine safety management center | 25% | Terminal qualification review, policy issuance, and state management |
| Device-manufacturer node | 20% | Device provenance, manufacturing information, and qualification evidence |
| Underground edge node | 25% in total; each node ≤10% | Regional authentication-record submission and consensus participation |

As shown in Table 1, regulatory nodes receive a relatively high initial weight because they perform compliance verification, historical-violation checking, and consensus veto. The mine safety management center is responsible for terminal qualification review, policy issuance, and state management; manufacturer nodes provide device-provenance and qualification evidence; and underground edge nodes submit regional authentication records and participate in consensus. To prevent a single edge node from obtaining excessive selection probability, its individual weight is limited to 10%. These values are initial design parameters and should be adjusted according to the deployed node scale, trust model, and regulatory requirements.

The system defines five identifier states: inactive, normal, warning, frozen, and revoked. A terminal transitions from inactive to normal after qualification review and on-chain registration. It enters the warning state when environmental factors reach a safety threshold, with high-risk permissions restricted. Expired qualifications, data tampering, or three consecutive authentication failures cause the terminal to be frozen. A warning that persists for 1 h also triggers automatic freezing. Revocation can be initiated only by a regulatory node or mine-management node and is irreversible.

## 3.3 Fog-node self-organization and local authentication

After registration results and terminal states have been recorded on-chain, frequent authentication that continues to depend on blockchain or cloud interaction would be unable to satisfy underground real-time and continuity requirements. The system therefore moves online authentication and failover to the fog layer. Fog nodes form a self-organizing network over 433 MHz Long Range (LoRa) communication [24]. When a node fails, a neighboring node first verifies its on-chain takeover authorization, obtains the corresponding authentication material through a secure channel, derives the required direction keys, and assumes the authentication task. During disconnection, only a preauthorized node holding a valid backup credential may perform minimum-permission authentication. The proposed communication configuration uses a 50 m coverage radius, a 125 kHz bandwidth, and spread-spectrum modulation to improve interference resistance.

The fixed core terminal–fog hop is considered separately as a reference 100-Mb/s full-duplex industrial Ethernet link conforming to IEEE 802.3 [25]; it was not deployed or physically measured. The 433-MHz LoRa configuration above pertains only to a proposed fog-to-fog

coordination path, not the three-message core-terminal authentication path. Wireless ordinary-terminal configurations require separate radio airtime budgets and are outside the present experiment.

Core terminals undergo full-field and environmental-consistency verification. Ordinary terminals may reuse identity states, static-field results, and environmental references that have remained anomaly-free for 1 h to reduce on-chain queries and attribute parsing. Nevertheless, every new session must still generate $N_T$, $N_F$, $IV_1$, and $IV_2$, execute $M_1$-$M_3$, and derive fresh keys. The $E_T^t$ contained in $M_1$ is used to recompute the digest, while $Ref_T^b$ is used to retrieve a digest-verified historical vector. Neighborhood verification uses only terminals in the normal state and within the same sampling window. Any increase in risk, location change, or state-version update immediately invalidates the cache. Cache lifetimes are 30 min for core terminals and 1 h for ordinary terminals, and the target response time for core-terminal authentication is 8 ms.

For a transparent wired-link serialization illustration (not measured packet sizes), assume each of M1, M2 and M3 carries 64–512 B of application/transport payload plus an illustrative 38 B per-frame physical/MAC allowance. At 100 Mb/s, the three serializations together require 3×(64+38)×8/10^8 = 0.0245 ms to 3×(512+38)×8/10^8 = 0.1320 ms, before store-and-forward switching, queuing, database access, software processing and retransmissions. Wired Ethernet has no radio airtime. Since protocol field widths and actual frames were not captured in the software prototype, this interval is a stated reference assumption, not a measured end-to-end bound. The 8-ms single-core-terminal and 15-ms group targets are unverified design objectives and cannot be inferred from the batch turnaround values in the scalability evaluation of Section 5.3.

When a terminal moves across regions, the target fog node queries its historical state and environmental reference tuple using $ID_T^{\text{base}}$. After on-chain authorization, the mine safety management center derives and securely delivers the target-node-specific $K_{TF}$. The target node derives the two direction-specific message-protection keys using the same purpose labels and then re-executes the three-message authentication, in which $M_{2a}$ and $M_{2b}$ are the two logical components of the second network message, and $M_3$ provides terminal confirmation. Group authentication in high-risk working areas has the highest priority, with a target latency of 15 ms.

## 3.4 Smart-contract-based dynamic permission control

Fog-node authentication confirms whether a terminal is eligible to access the system, but it does not determine which operations the terminal may perform under different risk conditions. The system therefore uses smart contracts to construct a three-dimensional association matrix among terminal security level, environmental risk level, and operator permission [26]. Terminal security is classified as core, important, or ordinary; environmental risk is classified as low, medium, high, or extreme; and personnel permissions are categorized as management, operation, or viewing. Core terminals in an extreme-risk environment retain only management and emergency-operation permissions, ordinary terminals in a high-risk environment retain only viewing permission, and important-terminal permissions decrease progressively as risk increases.

The permission-policy design considers persistent packet loss, unauthorized relocation, reduced on-line stability, and environmental deviation from a digest-verified historical vector retrieved from controlled off-chain storage. A reference mean, if used, must be calculated from authenticated off-chain historical samples rather than claimed to be stored on-chain. The numerical triggering limits given in preliminary examples (including 5% packet loss and 30% environmental deviation) are uncalibrated configuration examples, not experimentally validated field thresholds. Every proposed permission adjustment is subject to identity, terminal-state and policy checks and is intended to generate an auditable event; smart-contract enforcement itself was not implemented in this study.

## 3.5 Cross-chain collaboration, anomaly detection, and traceability audit

Dynamic permission control restricts the operations available at the authentication instant, but terminal conditions and underlying risks continue to change. The system must therefore monitor post-authentication behavior continuously and incorporate anomaly-handling results into traceability auditing. After authentication and permission assignment, environmental, operating-state, terminal-identity, and location-related features within a continuous time window are arranged as a two-dimensional time-step-by-feature matrix and supplied to the CNN–LSTM model [11]. The input is first normalized using min–max normalization:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min} + \varepsilon} \tag{16}$$

In Eq. (16), $x$ and $x'$ denote the original and normalized input features; $x_{min}$ and $x_{max}$ are the minimum and maximum values of the feature in the training data; and ε is a small positive constant that prevents division by zero.

In the newly rerun controlled dataset, feature screening is calculated from the training partition only. For each of the eight input features, five sequence summaries (mean, minimum, maximum, standard deviation, and last-10-step minus first-10-step mean) are evaluated for mutual information with the four-class target using a three-neighbor estimator. The largest mutual information of the five summaries is divided by the largest value across the eight features; a normalized score of at least 0.3 retains the feature. Seven features are selected, and humidity is

excluded. The standalone CNN baseline uses three 3×3 convolutional layers (16, 32, and 64 output channels), ReLU activation, temporal 2× pooling after the first two blocks, global average pooling, and a four-class linear classifier. The LSTM has two 64-unit recurrent layers; CNN–LSTM feeds features extracted by the same convolutional blocks into a two-layer 64-unit LSTM after projection. Class probabilities are modeled by Softmax and the weighted cross-entropy loss below.

$$p_c = \frac{\exp(z_c)}{\sum_{j=1}^{C} \exp(z_j)} \tag{17}$$

$$\mathcal{L} = -\frac{1}{N_s} \sum_{i=1}^{N_s} \sum_{c=1}^{C} \alpha_c \, y_{i,c} \ln p_{i,c}, \alpha_c = \begin{cases} 2, & c \in C_{\text{high}}, \\ 1, & c \notin C_{\text{high}}. \end{cases} \tag{18}$$

In Eq. (17), $p_c$ is the predicted probability of class $c$; $z_c$ is the corresponding network output; $C$ is the number of anomaly classes; and $j$ is the class-summation index. In Eq. (18), $L$ is the weighted cross-entropy loss; $N_s$ is the number of training samples; $i$ is the sample index; $y_{i,c}$ and $p_{i,c}$ are the ground-truth label and predicted probability; $\alpha_c$ is the class weight; and $C_{\text{high}}$ is the set of high-risk anomaly classes.

The new transparent generator produces 4000 independently synthesized 60-step sequences with eight original signals. Its fixed index partition contains 2800 training, 800 validation, and 400 testing samples; it is not an underground chronological-field split. Training-only screening and range normalization prevent test-label leakage. CNN, LSTM, and CNN–LSTM use Adam (learning rate 0.001), batch size 32, up to 500 optimization steps, and class weights (1,1,2,2), selecting checkpoints from validation loss. The dataset generator seed is 640807 and training seeds are 2026, 2027 and 2028. Class rules, proportions, AR(1) noise (coefficient 0.65), delayed event injection, nuisance drift and full model implementations are provided in the reproducibility supplement. Field validation is beyond the present study.

Anomalies are classified as general, severe, or emergency events. A general anomaly triggers re-authentication and cache fallback; a severe anomaly freezes permissions and issues an alarm; and an emergency anomaly freezes permissions, prioritizes the emergency channel, and issues an on-site equipment shutdown warning. The 80-ms emergency-response target is an unverified design objective; actuator, scheduling, communication and alarm delays were not measured. Fog nodes perform incremental backup every 10 min and retain core authentication information, the three most recent results, and basic permission rules for 24 h.

By integrating dynamic permission computation, continuous anomaly monitoring, and graded response, the closed-loop for permission adjustment, anomaly handling, and cross-chain traceability is shown in Figure 3.

As shown in Figure 3, terminal security level $L_T$, environmental risk level $R_t$, operator role $U$, and current terminal state $S_T$ are used as inputs to smart-contract-based permission computation. An invalid permission request is rejected and recorded, whereas a valid request enters continuous CNN-LSTM-based monitoring. General, severe, and emergency anomalies trigger re-authentication with cache fallback, permission freezing with an alert, and permission freezing with emergency-channel priority and a shutdown warning, respectively. The response updates permissions and the five-state terminal status. Audit events are organized through a Merkle mechanism and desensitized audit data are synchronized from the enterprise private blockchain to the regulatory consortium blockchain. Regulatory feedback then updates local policies and subsequent permission decisions, completing the control loop.

Cross-chain interoperability enables verifiable information exchange among heterogeneous blockchain systems [27,28]. To enable the regulatory side to verify authentication and anomaly records generated inside the enterprise domain, the system further establishes a cross-chain collaboration mechanism between the enterprise private blockchain and the regulatory consortium blockchain. A unified authentication-information protocol is used for cross-chain interaction, with terminal identifier, security level, state type, authentication timestamp, and verification code as the core fields. Manufacturer trade secrets are removed before information is shared with the regulatory chain; only regulatory fields are retained, and access is controlled by smart contracts [29].

The cross-chain permission-filtering and audit synchronization described here are conceptual workflows rather than deployed and evaluated smart contracts or chain-to-chain transfers.

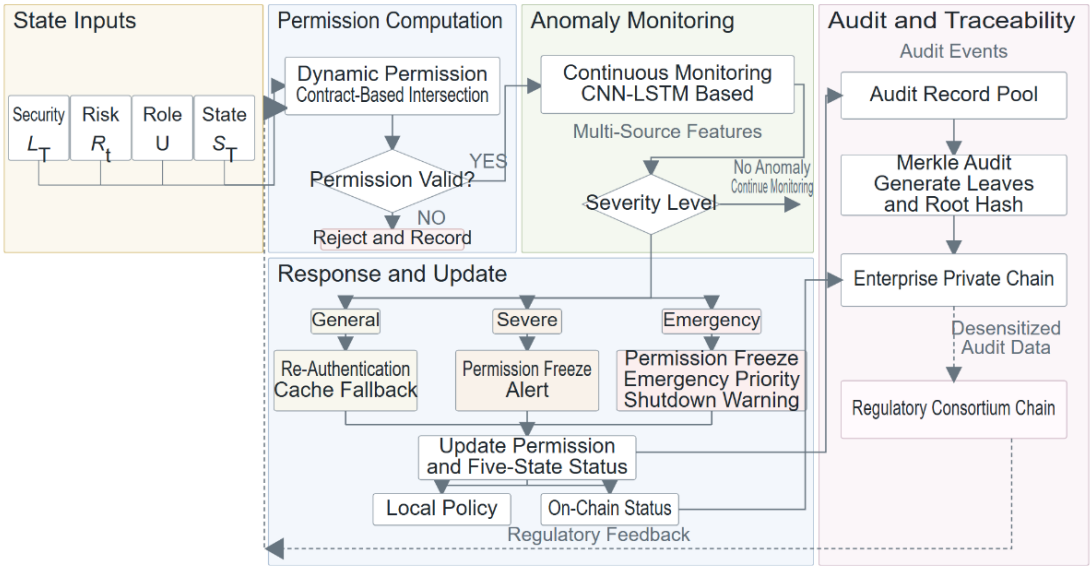


**Figure 3.** Dynamic permission adjustment, anomaly response, and cross-chain traceability

Event records generated by authentication, authorization, and anomaly handling are submitted to the traceability-audit module. Individual authentication records, permission-adjustment records, and anomaly-handling records form the leaves of a Merkle tree [30]. The leaf, parent, and root nodes are defined as

The Merkle hashing principle is grounded in the original hash-tree authentication construction [31].

$$h_i^{(0)} = \mathrm{H}[\mathrm{Encode}(r_i)] \tag{19}$$

$$h_i^{(l+1)} = \mathrm{H}\left(h_{2i}^{(l)} \parallel h_{2i+1}^{(l)}\right) \tag{20}$$

$$R_{\mathrm{Merkle}} = h_0^{(L)} \tag{21}$$

In Eqs. (19)-(21), $r_i$ is the ith authentication, permission-adjustment, or anomaly-handling record; $h_i^0$ is its leaf hash; $l$ is the Merkle-tree level; $h_i^{(l+1)}$ is a parent hash generated from adjacent child nodes; "‖" denotes concatenation; $R_{\mathrm{Merkle}}$ is the Merkle root; and $L$ is the highest tree level.

In the unimplemented cross-chain audit design, once either 100 records have accumulated or 30 min have elapsed, the Merkle root hash and core-field hashes would be synchronized to the regulatory consortium blockchain. Records can be queried by terminal identifier, time range, region, behavior type, and risk level, enabling the access events, permission operations, and anomalous behavior of terminals related to an accident to be traced by location and time.

## 4. Formal security analysis

This section applies Burrows–Abadi–Needham (BAN) logic to the three-message authentication protocol described in Section 2.3 [32]. BAN logic characterizes the beliefs that protocol participants can establish after receiving protected messages and is suitable for analyzing identity authentication, message freshness, and session-key confirmation. The analysis proceeds through notation and inference rules, protocol idealization, initial assumptions, security goals, formal derivation, and supplementary attack-resistance analysis.

### 4.1 BAN logic notation and inference rules

The inference rules follow the original BAN-logic formulation [33]. To standardize the notation used in protocol idealization, security-goal specification, and BAN-logic derivation, Table 2 lists the principal BAN symbols adopted in this paper.

In Table 2, $P$ and $Q$ denote protocol principals, $X$ and $Y$ denote messages or propositions, and $K$ denotes a secure key shared by two principals. The notation describes belief, message reception, prior utterance, jurisdiction, freshness, shared-key relationships, and key-protected messages. These symbols are used in the inference rules and subsequent security derivation.

The main inference rules used in this paper are as follows. Message-meaning rule:

$$\frac{P \mid\equiv P \overset{K}{\leftrightarrow} Q, \quad P \lhd \{X\}_K}{P \mid\equiv Q \mid\sim X} \tag{22}$$

Freshness-concatenation rule:

$$\frac{P \mid\equiv \#(X)}{P \mid\equiv \#(X, Y)} \tag{23}$$

Nonce-verification rule:

$$\frac{P \mid\equiv \#(X), \quad P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X} \tag{24}$$

Jurisdiction rule:

$$\frac{P \mid\equiv Q \Rightarrow X, \quad P \mid\equiv Q \mid\equiv X}{P \mid\equiv X} \tag{25}$$

Eqs. (22)-(25) are, respectively, the message-meaning, freshness-concatenation, nonce-verification, and jurisdiction rules. The meanings of $P$, $Q$, $X$, $Y$, and $K$ are consistent with Table 2; the expressions above each inference bar are premises, and the expression below the bar is the conclusion.

Table 2. BAN logic notation

| Symbol | Meaning |
|---|---|
| $P \mid\equiv X$ | Principal P believes proposition X |
| $P \lhd X$ | Principal P has received and can read message X |
| $P \mid\sim X$ | Principal P once said or expressed X |
| $P \Rightarrow X$ | Principal P has jurisdiction over proposition X |
| $\#(X)$ | Message or proposition X is fresh |
| $P \overset{K}{\leftrightarrow} Q$ | P and Q share the secure key K |
| $\{X\}_K$ | Message X is protected by key K |

## 4.2 Authentication protocol and idealized messages

Section 2.3 describes the authentication process at the workflow level as three messages, $M_1$, $M_2$, and $M_3$. This subsection specifies the direction keys, session key, associated data, nonces, timestamps, and initialization vectors and then establishes the corresponding BAN idealization. Let $T$ denote a terminal and $F$ its currently authorized fog node. The parties obtain the long-term authentication key $K_{TF}$ through the domain-key mechanism described in Section 2.3. To prevent one key from being reused across communication directions or cryptographic purposes, two independent SM4-GCM message-protection keys and the current session key are first derived:

$$C = ID_T^{\text{base}} \parallel ID_F \parallel v_K,$$
$$K_{\text{enc}}^{T\to F} = \text{KDF}(K_{TF}, \text{AUTH-ENC-TF} \parallel C),$$
$$K_{\text{enc}}^{F\to T} = \text{KDF}(K_{TF}, \text{AUTH-ENC-FT} \parallel C),$$
$$SK = \text{KDF}(K_{TF}, \text{SESSION} \parallel N_T \parallel N_F \parallel ID_{\text{mix}}^t \parallel C). \tag{26}$$

Eq. (26) jointly defines the derivation context and key schedule. In Eq. (26), $C$ denotes the key-derivation context; $K_{\text{enc}}^{(T\to F)}$ and $K_{\text{enc}}^{(F\to T)}$ are the SM4-GCM message-protection keys for terminal-to-fog and fog-to-terminal traffic; AUTH-ENC-TF, AUTH-ENC-FT, and SESSION are purpose-separation labels; $N_T$ and $N_F$ are fresh nonces generated by the two parties; and $SK$ is the session key.

The first message $M_1$ is generated by the terminal to submit identity, environmental, and freshness information. Its associated data, plaintext payload, authenticated-encryption output, and complete network message are defined as follows:

$$AD_1 = \text{Encode}(\text{Ver}, \text{Type}_1, ID_T^{\text{base}}, ID_F, v_K),$$
$$P_1 = (ID_{\text{mix}}^t, E_T^t, N_T, t_T),$$
$$(C_1, Tag_1) = \text{SM4-GCM.Enc}_{K_{\text{enc}}^{T\to F}}(IV_1, P_1, AD_1),$$
$$M_1 = (AD_1, IV_1, C_1, Tag_1). \tag{27}$$

Eq. (27) defines the construction of the first network message. In Eq. (27), $AD_1$ is the associated data of the first message; Ver and $Type_1$ are the protocol version and message type; $P_1$ is the plaintext payload; $C_1$ and $Tag_1$ are the SM4-GCM ciphertext and authentication tag; $IV_1$ is the unique terminal-to-fog initialization vector; and $M_1$ is the complete authentication request.

The second message M2 is generated after the fog node has verified the identity state, permission policy, and environmental consistency. It returns the terminal state, fog-node nonce, and fog-node key-confirmation value:

$$AD_2 = \text{Encode}(\text{Ver}, \text{Type}_2, ID_T^{\text{base}}, ID_F, v_K),$$
$$P_2 = (ID_F, N_T, N_F, S_T, t_F),$$
$$(C_2, Tag_2) = \text{SM4-GCM.Enc}_{K_{\text{enc}}^{F\to T}}(IV_2, P_2, AD_2),$$
$$M_2' = (AD_2, IV_2, C_2, Tag_2). \tag{28}$$

$$Tag_F = \text{HMAC}_{SK}(\text{Fconf} \parallel H(M_1 \parallel M_2')),$$
$$M_2 = M_2' \parallel Tag_F \tag{29}$$

Eq. (28) defines the authenticated response body of the second message. In Eq. (28), $AD_2$, $P_2$, $C_2$, $Tag_2$, and $IV_2$ are the associated data, plaintext payload, ciphertext, authentication tag, and unique initialization vector of the second message; $M_2'$ is the authenticated-encryption response body. Eq. (29) defines the fog-node confirmation

value and the complete response; $M_2$ is the complete response after appending the fog-node confirmation value. In Eq. (29), $Tag_F$ is the key-confirmation value generated with $SK$, Fconf is the fog-node confirmation label, and H($M_1$ ‖ $M_2$) is the transcript digest.

The third message $M_3$ is generated after the terminal has verified $M_2$ and explicitly demonstrates that the terminal has obtained the current session key:

$$M_3 = \mathrm{HMAC}_{SK}\big(\mathrm{Tconf} \parallel N_F \parallel \mathrm{H}(M_1 \parallel M_2)\big) \quad (30)$$

In Eq. (30), $M_3$ is the terminal explicit key-confirmation message; Tconf is the terminal-confirmation label; $N_F$ binds the response nonce; and H($M_1$ ‖ $M_2$) binds the complete protocol transcript.

Both $AD_1$ and $AD_2$ contain the protocol version, message type, stable base identifier, fog-node identifier, and key version. Consequently, fields that need not be encrypted but must be authenticated are cryptographically bound to the ciphertext. $M_1$ uses $K_{\mathrm{enc}}^{T\to F}$ and the direction-unique $IV_1$, while $M_2$ uses $K_{\mathrm{enc}}^{F\to T}$ and the direction-unique $IV_2$. Each sending direction employs a persistent counter or an equivalent mechanism that guarantees IV uniqueness and prevents reuse under the same direction key. The actual protocol still exchanges three network messages: $M_{2a}$ corresponds to the SM4-GCM response body of $M_2$, whereas $M_{2b}$ corresponds to the key-confirmation semantics carried by $Tag_F$ in the same message. Therefore, BAN idealization separates $M_2$ into two logical components without introducing an additional round trip. The terminal request is idealized first:

$$M_1\colon\ T \to F\colon \quad (31)$$
$$\{ID_T^{\mathrm{base}}, ID_{\mathrm{mix}}^{t}, E_T^{t}, N_T, t_T\}_{K_{\mathrm{enc}}^{T\to F}}.$$

The second network message is represented by an authenticated-response component and a key-confirmation component:

$$M_{2a}\colon\ F \to T\colon \{N_T, N_F, ID_F, S_T, t_F\}_{K_{\mathrm{enc}}^{F\to T}}. \quad (32a)$$

$$M_{2b}\colon\ F \to T\colon \{\mathrm{Fconf}, N_T, N_F, H\big(M_1 \parallel M_2^{'}\big), T \overset{SK}{\leftrightarrow} F\}_{SK}. \quad (32b)$$

The terminal's explicit key-confirmation message is idealized as follows:

$$M_3\colon\ T \to F\colon \{\mathrm{Tconf}, N_F, H(M_1 \parallel M_2), T \overset{SK}{\leftrightarrow} F\}_{SK}. \quad (33)$$

Eqs. (31), (32a), (32b), and (33) respectively represent the BAN idealizations of the three network messages. The notation $X_\mathrm{K}$ indicates that message $X$ is protected by key $K$; $M_{2a}$ and $M_{2b}$ denote the authenticated-encryption body and key-confirmation semantics of the second network message; and the bidirectional arrow between terminal $T$ and fog node $F$ denotes their shared session key $SK$. This logical separation is used only for the security proof and does not add a network round trip.

## 4.3 Initial assumptions

The BAN derivation relies on initial beliefs concerning long-term-key trust, authentication-parameter freshness, and agreement on the locally derived session key. These assumptions describe the beliefs accepted at the start of authentication and do not, by themselves, prove that the peer actually possesses the current session key.

(i) Long-Term-Key Trust Assumptions. Assumption A1, $F \mid\equiv T \overset{K_{\mathrm{enc}}^{T\to F}}{\leftrightarrow} F$, states that the fog node believes the terminal-to-fog message-protection key is known only to T and F. Assumption A2, $T \mid\equiv T \overset{K_{\mathrm{enc}}^{F\to T}}{\leftrightarrow} F$, states that the terminal holds the corresponding belief for the fog-to-terminal key. These assumptions provide the key-trust basis for inferring the sender of a protected message.

(ii) Authentication-Parameter Freshness Assumptions. Assumption A3, F |≡ #($N_T$, t_T), is interpreted as a conditional belief reached only after the timestamp-window and replay-cache checks succeed, not as a priori freshness of an unchecked peer nonce. Assumptions A4, T |≡ #$N_T$, and A5, F |≡ #$N_F$, state that each party believes its locally generated nonce is fresh. These assumptions bind received messages to the current authentication session and support replay resistance.

Operational assumption A3 requires a trustworthy fog-node clock, an explicit acceptance window and a replay cache; it is only applied after those verifications. Thus F believing freshness of the received tuple cannot be used as an unconditional premise before receiving and validating M1.

(iii) Expected Session-Key Agreement Assumptions. Assumptions A6, $T \mid\equiv T \overset{SK}{\leftrightarrow} F$, and A7, $F \mid\equiv T \overset{SK}{\leftrightarrow} F$, state that after local key derivation each party believes that $SK$ is the expected shared key for the current session. They express consistency of the derivation context but do not prove peer possession. Possession is established through the explicit confirmation values carried by $M_2$ and $M_3$. The physical correctness of environmental measurements is not treated as a BAN jurisdiction assumption and is instead evaluated using the environmental-message digest, trusted history, and neighborhood consistency.

## 4.4 Security goals

The security goals are grouped into mutual authentication, bilateral session-key possession confirmation, and session-key freshness.

(i) Mutual-Authentication Goals. Goal G1 requires the fog node to believe that the stable base identifier, current hybrid identifier, environmental message, and freshness parameters were submitted by a legitimate terminal in the current session. Goal G2 requires the terminal to believe that the authenticated response body was generated by the currently authorized fog node in the same session.

(ii) Session-Key-Possession Goals. Goal G3 requires the terminal, after verifying the fog-node confirmation value, to believe that the fog node actually possesses $SK$. Goal G4 requires the fog node, after verifying the terminal explicit-confirmation message, to believe that the terminal possesses the same session key.

(iii) Session-Key-Freshness Goal. Goal G5 requires both parties to believe that the current session key is fresh. This goal is supported by the computational property of the KDF that binds both fresh nonces, the current hybrid identifier, and the identity context into $SK$; it does not follow solely from the BAN inference rules. BAN logic also does not establish the physical truth of environmental sensor values.

## 4.5 BAN logic derivation and session-key freshness analysis

Using the idealized messages, initial assumptions, and security goals defined above, the following derivation considers terminal-request authentication, the fog-node response and session-key possession, and terminal explicit key confirmation. It then analyzes computational freshness of the session key.

### 4.5.1 Fog-node authentication of the terminal request

The notation used in the following derivation is defined in Table 2, Eqs. (31), (32a), (32b), and (33), and Assumptions *A1-A7*; it is therefore not redefined in each subsequent step.

From $M_1$, Assumption $A_1$, and the message-meaning rule, we obtain

$$F \mid\equiv T \mid\sim \left(ID_T^{\text{base}}, ID_{\text{mix}}^t, E_T^t, N_T, t_T\right) \tag{34}$$

From $A_3$ and the freshness-concatenation rule, the fog node believes that the request belongs to the current session:

$$F \mid\equiv \#\left(ID_T^{\text{base}}, ID_{\text{mix}}^t, E_T^t, N_T, t_T\right) \tag{35}$$

Combining Eqs. (34) and (35) with the nonce-verification rule gives

The use of nonce verification in Eqs. (35)–(36) is conditional on the operational A3 timestamp-window and replay-cache checks having succeeded. BAN message-meaning reasoning alone does not establish that an arbitrary received nonce was fresh or that its measured environmental payload was physically correct.

$$F \mid\equiv T \mid\equiv \left(ID_T^{\text{base}}, ID_{\text{mix}}^t, E_T^t, N_T, t_T\right) \tag{36}$$

Eq. (36) directly satisfies $G_1$. It shows that the fog node believes the stable base index, current hybrid identifier, and environmental message were submitted by the terminal in the current session. The physical reliability of the environmental values must still be evaluated using the digest of $E_T^t$, the off-chain historical reference, and the same-window neighborhood-consistency checks described in Section 3.1.

### 4.5.2 Fog-node response and session-key possession

From $M_{2a}$, Assumption $A_2$, and the message-meaning rule, we obtain

$$T \mid\equiv F \mid\sim (N_T, N_F, ID_F, S_T, t_F) \tag{37}$$

Because $M_{2a}$ echoes the terminal-generated fresh nonce $N_T$, Assumption $A_4$ and the freshness-concatenation rule yield

$$T \mid\equiv \#(N_T, N_F, ID_F, S_T, t_F) \tag{38}$$

Applying the nonce-verification rule to Eqs. (37) and (38), the terminal confirms that the response body was generated by a legitimate fog node in the current session:

$$T \mid\equiv F \mid\equiv (N_T, N_F, ID_F, S_T, t_F) \tag{39}$$

Eq. (39) satisfies $G_2$. The terminal then verifies $M_{2b}$ with its locally derived $SK$. By $A_6$, the message-meaning rule, and freshness of $N_T$, (40) follows and satisfies $G_3$:

$$T \mid\equiv F \mid\equiv \left(T \overset{SK}{\leftrightarrow} F\right) \tag{40}$$

This result shows that the terminal not only authenticates the source of the fog-node response, but also confirms that the fog node actually possesses the session key derived in Eq. (26).

### 4.5.3 Terminal Explicit Key Confirmation

The fog node derives $SK$ according to Eq. (26) and therefore, by $A_7$, believes that it is the expected shared key for the current session. From $M_3$ and the message-meaning rule, we obtain

$$F \mid\equiv T \mid\sim \left(T \overset{SK}{\leftrightarrow} F\right) \tag{41}$$

Combining freshness of $N_F$ from $A_5$ with the nonce-verification rule gives

$$F \mid\equiv T \mid\equiv \left(T \overset{SK}{\leftrightarrow} F\right) \tag{42}$$

Because only an entity that possesses $SK$ can generate the correct terminal confirmation value, Eq. (42) satisfies $G_4$. The session key is jointly derived from the long-term authentication key, two fresh nonces, the current hybrid identifier, and the identity context. Therefore,

$$\#(N_T) \wedge \#(N_F) \Rightarrow \#(SK) \tag{43}$$

Eq. (43) expresses computational session-key freshness. Under declared trust and operational A3 freshness checks, the BAN derivation supports *G1–G4*, whereas *G5* follows from the computational property of the session-key KDF, which binds $N_T$, $N_F$, the current hybrid identifier, and the identity context. Accordingly, the analysis gives conditional support for mutual authentication and bilateral key-possession confirmation; it does not establish security under all device-compromise or network conditions. Because $SK$ is still derived from the long-term authentication key $K_{TF}$, the protocol does not claim forward secrecy after long-term-key compromise. The derivation covers authentication, message freshness, and key confirmation; threats such as denial of service, Sybil attacks, and privacy leakage require supplementary analysis at the system-mechanism level.

## 4.6 Resistance to other attacks

Beyond the authentication and key-confirmation properties established by the BAN analysis, the proposed system should also resist common attacks arising in underground communication and blockchain-assisted terminal access. Table 3 summarizes the corresponding threats and resistance mechanisms.

Overall, the scheme combines nonce- and timestamp-based freshness protection, direction-separated authenticated encryption, HMAC-based bilateral key confirmation, proposed permissioned consensus and tamper-evident audit mechanisms. The protocol mechanisms address replay, impersonation and message tampering within the stated key-security and freshness assumptions; Sybil resilience, availability and privacy protections of the unimplemented consensus and audit modules remain design-level claims. The present assessment is qualitative; resistance under large-scale denial-of-service attacks and partial node compromise remains to be evaluated on a prototype platform.

Table 3. Resistance mechanisms against representative attacks

| Attack | Resistance mechanism |
|---|---|
| Replay attack | $N_T$, $N_F$, and timestamps bind each session, and the fog node maintains a nonce replay cache; an old $M_1$ or $M_2$ cannot pass freshness verification. |
| Impersonation attack | Both the stable base identifier and current hybrid identifier are authenticated. Without $K_{TF}$, an adversary cannot derive the direction-specific message-protection keys or $SK$, and therefore cannot construct a valid SM4-GCM message or HMAC confirmation value. |
| Man-in-the-middle attack | $AD_1$ and $AD_2$ bind both identities, message direction, and key version. $M_{2b}$ and $M_3$ provide bilateral key-possession confirmation. Tampering with associated data, ciphertext, or an initialization vector causes SM4-GCM or HMAC verification to fail. |
| Sybil attack | Permissioned-node admission, PoS weight constraints, and the upper bound on the weight of a single edge node are intended to reduce the ability of forged identities to influence committee selection. |
| Data tampering | The environmental message $E_T^t$ is protected by SM4-GCM, and its SHA-256 digest, on-chain record, and Merkle proof jointly verify the integrity of data and audit records. |
| Denial of service / node failure | The design includes fog-node failover, local caching, and 24-h redundant backup to maintain basic authentication services, but these mechanisms were not implemented in the measured prototype. |
| Privacy leakage | The design combines zero-knowledge proofs, differential privacy, field masking, and authorized access to reduce exposure of sensitive information; these privacy protections remain design-level mechanisms in the present study. |

## 5. Analytical and simulation-based evaluation

This section evaluates the proposed scheme through analytical and simulation-based results. Functional and protocol-efficiency analyses are followed by scalability and ablation experiments that quantify the effects of submitted-request batch size, fog-layer parallel processing, and critical-path exclusion of a proposed ledger-recording step. The CNN-LSTM-based anomaly-detection module is then evaluated using a controlled synthetic dataset and compared with standalone CNN and LSTM models.

### 5.1 Functional comparison of authentication architectures

Before evaluating runtime scalability, the architectural characteristics of the proposed scheme are compared with centralized and typical blockchain-based authentication [5]. Table 4 summarizes the differences in identity basis, environmental awareness, central-point dependence, terminal differentiation, dynamic permission control, cross-region authentication, and accountability.

Table 4. Functional comparison of representative authentication architectures

| Dimension | Centralized authentication | Typical blockchain authentication | Proposed scheme |
|---|---|---|---|
| Identity basis | Central database | Static on-chain identity | Static-dynamic hybrid identifier |
| Environmental awareness | None or weak | Usually excluded from identity authentication | Environmental digest and numerical-consistency verification |
| Central single point of failure | Present | Distributed ledger, but the authentication entry may remain centralized | Local caching in the measured prototype; planned fog-node self-organization/failover |
| Terminal differentiation | Uniform policy | Usually uniform on-chain verification | Full/lightweight identifiers and hierarchical authentication |
| Dynamic permission | Central rules | Smart-contract rules | Terminal-environment-personnel association (design mechanism) |
| Cross-region authentication | Re-registration or manual synchronization | Dependent on on-chain queries | Historical-state retrieval and re-authentication |
| Accountability and traceability | Central logs | On-chain records | Merkle-based audit with planned cross-chain synchronization |

As shown in Table 4, centralized authentication relies primarily on a central service and is therefore susceptible to central-point failure. Typical blockchain-based authentication improves record immutability through distributed ledgers, but its authentication entry and terminal-processing policy may still lack environmental awareness and differentiated processing. The proposed scheme combines static-dynamic hybrid identifiers, fog-node local authentication, smart-contract-based dynamic permission control, and Merkle-based cross-chain auditing, thereby extending authentication from identity verification to environment-aware access control and traceability. The runtime scalability of these architectural alternatives is quantitatively evaluated in the subsequent simulation experiments.

The functional comparison describes the mechanisms covered by each architecture but does not directly indicate runtime cost. The protocol is therefore further examined in terms of communication rounds and cryptographic computation.

Table 4 compares the intended architectural functions, not end-to-end deployed capabilities. In particular, the

permission-contract, fog-failover and cross-chain features are not included in the measured authentication prototype.

## 5.2 Protocol efficiency analysis

The three-message authentication protocol limits the online interaction between a terminal and its authorized fog node to $M_1$, $M_2$, and $M_3$. During each session, the terminal generates the authentication request, the fog node verifies the current environmental message and trusted reference information before generating the response, and the terminal subsequently provides explicit session-key confirmation. At each party, the principal cryptographic workload consists of approximately three KDF operations, one SM4-GCM encryption, one SM4-GCM decryption, two HMAC operations, and several hash operations. Cache reuse reduces repeated blockchain queries and static-attribute parsing, but does not omit current environmental verification, fresh-nonce generation, or bilateral session-key confirmation.

HMAC follows RFC 2104 [34]. For the proposed online exchange, the communication count is exactly three terminal–fog messages (M1, M2 and M3); without fixed encoded field lengths, the corresponding byte cost is B_M1+B_M2+B_M3, where each B includes the transmitted fields, IV and tag. At the protocol-operation level, the processing path contains KDFs, one SM4-GCM encryption and decryption at each message direction, HMAC confirmation and SHA-256/environmental checks. An exact universal byte count would be unjustified before encoding and environment-message sizes are fixed. Closely related IIoT or coal-mine-oriented authentication studies [5,6,8] mainly report communication rounds, security mechanisms, or operation classes rather than a normalized byte count under a common field-length model, so they are used here as directly relevant architectural comparators. For additional quantitative context, two published blockchain-assisted IoT studies report 1052 and 2368 bits under their separate schemes [35,36]. These are source-reported costs from different protocols and security models; they are not head-to-head benchmarks against a byte-undefined industrial prototype. Existing IIoT studies [5,6,8] address decentralized or edge authentication but their execution environments cannot be directly equated to our single-host turnaround metric.

As an implementation-specific byte-accounting example (NOT a production message format), the archived in-process prototype allocates 124 B to the concatenated M1 fields (16-B associated data, 12-B IV, 80-B ciphertext and 16-B tag), 114 B to M2 (16-B associated data, 12-B IV, 38-B ciphertext, 16-B tag and 32-B HMAC), and 32 B to M3 (HMAC), totaling 270 B/2160 bits before any network stack overhead. These fields were passed within a single process and never serialized or sent as Ethernet packets. The test invocation performs six KDF calls, two SM4-GCM encryptions, two decryptions and four confirmation-generation/verification HMACs (not counting PRFs inside the KDF); therefore 2160 bits is a transparent software-field illustration, not a deployment-validated wire-size guarantee. Cross-paper 1052-bit and 2368-bit literature figures use different protocol components and cannot establish a numerical advantage.

An archived, separately timed cryptographic microbenchmark of this three-message software implementation (Intel Xeon Platinum 8370C; Python 3.13.5) recorded a mean local cryptographic processing time of 225.924 μs, excluding transmission, queuing, database access and ledger operations (supplementary benchmark report). Related IIoT/edge studies [5,6,8] emphasize functional architecture and security properties, but do not publish directly reusable common-platform primitive timings for this protocol shape. For additional source-reported computation context, Kuang et al. measured approximately 257.401 μs for their blockchain/lattice IoT authentication under a 100-bit security configuration [35], while Kim et al. reported a total of 0.0380 ms for their four-message blockchain-assisted IoT protocol using their own OpenSSL primitive timings [36]. Together with the communication figures of 1052, 2368 and the illustrative 2160 bits discussed above, these provide quantitative context, not a common-platform efficiency ranking: workloads, implementation assumptions, hardware, security objectives and fields differ. The benchmark script and archived timing results are provided in Supplementary Material S3.

Registration is designed to require additional interactions for policy checks and ledger admission, but neither PoS committee selection nor full PBFT consensus was implemented in the present benchmark. Routine online authentication instead uses the local fog processing path, and the proposed architecture places future ledger recording outside the immediate request-response critical path. The prototype times processing with that record step omitted; it does not demonstrate a working asynchronous chain-writer, transaction persistence or failure recovery. Consequently the following measurements should be interpreted as controlled software workload comparisons, not live ledger throughput or underground communication latency.

## 5.3 Scalability evaluation

A single-host Python multiprocessing experiment compared a centralized reference, a simplified PBFT-style local emulator and a four-worker fog prototype at submitted-request batch sizes of 100, 500, 1000 and 2000, each repeated in five independent runs. The workload generator submitted all requests for each batch to a process executor; the batches do not represent 100–2000 simultaneously executing CPU tasks. The centralized configuration used one worker, SQLite lookup and HMAC. The PBFT-style comparator used one worker, four in-memory logical validators, a three-vote quorum and prepare/commit-like hash/HMAC checks; it exchanged no messages over a real inter-node network, applied no external propagation-delay model and did not commit to a

distributed ledger. The proposed configuration used four workers, SQLite, environmental-distance checks, KDF, SM4-GCM and HMAC-based key confirmation, with ledger recording excluded from the timed path. Request turnaround time was recorded from submission to collection by the controlling process, including process scheduling/queueing; throughput was calculated from completed requests per elapsed batch time. Five-run mean values and Student-t 95% confidence intervals are shown in Figure 4, while per-run/request data, request generator, software package versions and operating environment are provided in the supplement. The measured values exclude terminal–fog physical communication, actual 21-node consensus and chain persistence.

The archived August 2026 runs recorded Linux 6.18.35 (x86-64), Python 3.13.5, cryptography 46.0.4, an AMD EPYC 9V74 CPU model visible to the container, five visible logical CPUs and 5.93 GiB available memory. These are the archived container environment characteristics, not specifications for a physical underground device or the author's personal computer. The archived request-generator seed is 20260806; all 60 scenario-run summaries and 54,000 per-request records are retained in the reproducibility supplement. The archived request-level data, per-run summaries, environment configuration, and rerun scripts are provided in Supplementary Material S1.

Figure 4(a) shows descriptive changes in mean single-host request turnaround time with submitted batch size. At 2000 submitted requests, the centralized, proposed fog and PBFT-style configurations respectively averaged 105.84, 125.85 and 128.94 ms. The observed difference between the proposed and PBFT-style means was −3.09 ms (approximately 2.4%). Because the comparator and proposed configuration differ in authentication workload and worker counts, the number does not isolate PBFT consensus overhead. On the five run-level results, a two-sided Welch test gave p=0.6761 (95% difference interval, −19.78 to 13.60 ms); the difference was not statistically significant at α=0.05.

Figure 4(b) reports mean software throughput at 2000 submitted requests of 9958.96, 8185.56 and 8608.77 requests/s for centralized, PBFT-style and proposed configurations, respectively. The observed proposed-versus-PBFT difference was +423.21 requests/s (approximately +5.2%), but a two-sided Welch test across the five runs yielded p=0.4383 (95% mean-difference interval, −810.80 to 1657.23 requests/s). Therefore, the throughput difference likewise lacks statistical significance. Figure 4 has been redrawn with t-based 95% mean confidence intervals and no clipped PBFT uncertainty bar. Neither the observed difference nor the software-emulation throughput establishes superiority over a deployed PBFT network.

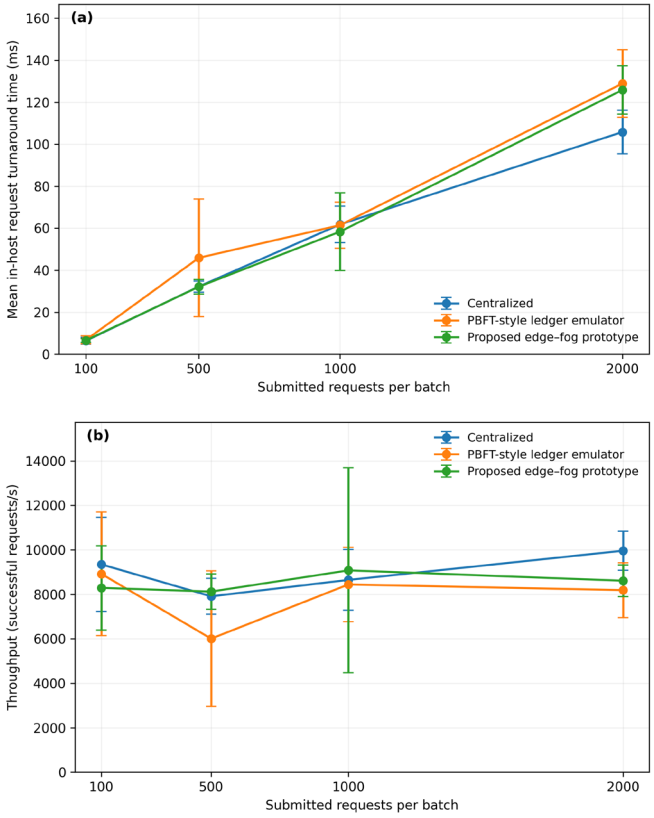


**Figure 4.** Single-host software prototype under increasing submitted-request batch size: (a) mean request turnaround time; (b) completed-request throughput
Error bars are five-run Student-t 95% confidence intervals; no physical link delay is included

## 5.4 Ablation study

A separate five-repeat batch of four ablation configurations tested the local fog-processing prototype at 2000 submitted requests per test: the complete four-worker path, one worker without fog parallelism, the four-worker path with a serial-lock local synchronous-recording emulator, and a one-worker path with the same emulator. This is a distinct batch from Section 5.3 and its absolute baselines should not be interchanged with the Figure 4 baseline. Table 5 now gives each configuration's run-averaged absolute turnaround time and throughput as well as within-batch percent changes. The simulated synchronous-recording penalty is not a measured full-network consensus latency; in particular, the +86.7% ablation change and the +2.4% cross-architecture observation are not estimates of the same incremental cost.

Table 5. Ablation study of the proposed scalability mechanisms at 2000 submitted requests

| Configuration | Latency (ms; change) | Throughput (req/s; change) |
|---|---|---|
| Full proposed scheme | 161.009; 0.0% | 7880.032; 0.0% |
| w/o fog parallelism | 266.716; +65.7% | 3964.020; −49.7% |
| w/o asynchronous recording | 300.591; +86.7% | 3505.417; −55.5% |
| w/o both mechanisms | 478.617; +197.3% | 2130.718; −73.0% |

Relative to the full configuration (161.009 ms; 7880.032 requests/s) within the original ablation batch, the one-worker setting increased mean time by 65.7%, the local synchronous-recording emulator increased it by 86.7%, and removing both mechanisms increased it by 197.3%; corresponding observed throughput reductions were 49.7%, 55.5% and 73.0%. As a further controlled diagnostic, we newly compared the same authentication workload with four workers both with and without a serial local recording-emulation step: across five runs, means changed from 186.55 to 300.78 ms and from 5658.95 to 3357.55 requests/s (61.2% time increase; 40.7% throughput decrease). This diagnostic comes from another software environment and is not merged with the historic Figure 4 or Table 5 values. These ablations quantify software-path effects only, not the reliability of a deployed asynchronous ledger.

## 5.5 Anomaly detection evaluation

The anomaly classifier was independently re-evaluated on a newly generated, fully documented controlled synthetic dataset; this is a replacement experiment, not exact recovery of the original undocumented training process. It comprises 4000 independent sequences of 60 time steps and eight initially generated features: methane, humidity, geological stress, location offset, packet-loss rate, on-line stability, terminal-state score and operating load. Normal (1957), general anomaly (571), severe anomaly (608) and emergency anomaly (864) classes were defined through an explicit generator (seed 640807) with sequence-level offsets, AR(1) time-correlated noise, class-independent nuisance variation and anomaly onset between time steps 10 and 53. The train-only mutual-information screening described in Section 3.5 retained seven features at 0.3. Fixed sample-index splits (2800/800/400) were used for training/validation/test, not chronological mine-field observations. The three model types each ran 500 optimization steps with seeds 2026, 2027 and 2028. All underlying samples, source code and per-class confusion matrices are supplied as reproducibility materials. The dataset generator, training scripts, seed-wise metrics, and confusion matrices are included in Supplementary Material S2.

Table 6 reports the mean ± sample standard deviation over three independent training seeds; Macro-Precision, Macro-Recall, and Macro-F1 are averaged uniformly over the four classes. All three models use the identical 400-sequence held-out test partition.

Table 6. Performance comparison of anomaly detection models

| Model | Accuracy | Precision | Recall | Macro-F1 |
|---|---|---|---|---|
| CNN | 0.626 ± 0.021 | 0.493 ± 0.016 | 0.525 ± 0.075 | 0.485 ± 0.038 |
| LSTM | 0.888 ± 0.006 | 0.893 ± 0.012 | 0.867 ± 0.010 | 0.876 ± 0.006 |
| **CNN-LSTM** | **0.910 ± 0.005** | **0.908 ± 0.014** | **0.906 ± 0.014** | **0.906 ± 0.009** |

General anomalies were generated by packet-loss and availability deterioration; severe anomalies by relocation and terminal-state deterioration; and emergency anomalies by methane/stress/load precursors. Some weak injected precursors remain below engineering alarm thresholds; synthetic class labels identify assigned scenario types rather than independently verified field events. The class imbalance and signal/noise distributions are generator settings, not estimated underground incidence rates. Training-only feature selection excluded humidity, and all three models used the same optimizer, class weighting and held-out test set. Representative four-class confusion matrices for CNN, LSTM and CNN–LSTM (seed 2026), with the complete nine-matrix count data, are provided in the accompanying supplement.

On this newly generated dataset, CNN–LSTM obtained mean accuracy 0.910±0.005, Macro-Precision 0.908±0.014, Macro-Recall 0.906±0.014 and Macro-F1 0.906±0.009. CNN obtained Macro-F1 0.485±0.038, whereas LSTM obtained 0.876±0.006. The CNN–LSTM mean was higher in these three seeded training trials, but no population-level superiority or statistical significance is established from this limited controlled sample. These results replace, rather than reproduce, the original Table 6 figures.

The revised experiment supports only the feasibility of the compared architectures for the stated synthetic classification rules. It does not establish real mine performance, robustness to unforeseen interference, or operating-level emergency-response time. The generator, classification process and all per-class error counts are available for inspection and future independent testing on field data.

## 6. Conclusion

This paper specifies a private-blockchain and edge–fog authentication framework for coal mine safety terminals with a three-message SM4-GCM/HMAC key-confirmation procedure and BAN-logic analysis under declared trust and freshness assumptions. PoS-weighted 21-node committee selection, failover, contract-enforced authorization and Merkle cross-chain audit remain design proposals rather than fully implemented subsystems. The evaluated modules are the single-host authentication-processing prototype and an independently generated controlled synthetic anomaly-classification study.

At a submitted batch size of 2000, the historical software benchmark observed 125.85 ms mean request turnaround and 8608.77 requests/s for the four-worker fog path, compared with 128.94 ms and 8185.56 requests/s for its simplified PBFT-style emulator. The corresponding descriptive differences (2.4% and 5.2%) were not statistically significant over five run-level trials (Welch $p=0.6761$ and $p=0.4383$). The separate software ablation and a same-workload diagnostic illustrate processing and serialization trade-offs but do not validate real network consensus. In the newly reproducible synthetic classifier test, CNN–LSTM obtained mean accuracy 0.910 and Macro-F1 0.906 over three seeds. Future evaluation must implement the proposed ledger and control services, characterize mine-specific terminal–fog links and 8/15/80-ms targets, and validate the anomaly rules on independent underground field data.

## Acknowledgements

This research was supported by the National Natural Science Foundation of China under Grant 52374154.

## Data and Code Availability

The source code and supporting data are provided in the accompanying Supplementary_Experimental_Materials.zip. Supplementary Material S1 (System_Audit_And_Reproduction.zip) contains archived authentication request-level records, five-run summaries, environment configuration, and separate rerun and diagnostic programs. Supplementary Material S2 (CNN_LSTM_Reproducible_Experiment.zip) contains the newly generated synthetic dataset, generator, training and evaluation scripts, and per-class confusion matrices. Supplementary Material S3 (Cryptographic_Microbenchmark_Source_And_Logs.zip) contains archived local cryptographic benchmarks and their source code. The archived system timings and the later synthetic-classifier results are separate controlled experiments; the package contains no underground field measurements or full distributed-ledger deployment traces.

## References

[1] Wang G, Ren H, Zhao G, Zhang D, Wen Z, Meng L, and Gong S. Research and practice of intelligent coal mine technology systems in China. International Journal of Coal

Science & Technology, 2022; vol. 9, Art. no. 24. doi: 10.1007/s40789-022-00491-3.
[2] Zhang J, Yan Q, Zhu X, and Yu K. Smart industrial IoT empowered crowd sensing for safety monitoring in coal mine. Digital Communications and Networks, 2023; vol. 9, no. 2, pp. 296–305. doi: 10.1016/j.dcan.2022.08.002.
[3] Ray Chowdhury A, Pramanik A, and Roy GC. IoT and LoRa based smart underground coal mine monitoring system. Microsystem Technologies, 2023; vol. 29, pp. 919–938. doi: 10.1007/s00542-023-05484-z.
[4] Pathak A, Al-Anbagi IS, and Hamilton HJ. Blockchain-enhanced zero knowledge proof-based privacy-preserving mutual authentication for IoT networks. IEEE Access, 2024; vol. 12, pp. 118618–118636. doi: 10.1109/ACCESS.2024.3450313.
[5] Cao Z, Wen X, Ai S, Shang W, and Huan S. A decentralized authentication scheme for smart factory based on blockchain. Scientific Reports, 2024; vol. 14, Art. no. 24640. doi: 10.1038/s41598-024-76065-x.
[6] Li Z, Liu X, Zhang N, and Wei Z. Blockchain-based certificateless cross-domain authentication scheme in the Industrial Internet of Things. Computers, Materials & Continua, 2024; vol. 80, no. 3, pp. 3835–3854. doi: 10.32604/cmc.2024.053950.
[7] Alzoubi YI, Al-Ahmad A, and Kahtan H. Blockchain technology as a fog computing security and privacy solution: An overview. Computer Communications, 2022; vol. 182, pp. 129–152. doi: 10.1016/j.comcom.2021.11.005.
[8] Zhu W, Chen X, and Jiang L. A secure and efficient authentication key agreement scheme for industrial internet of things based on edge computing. Alexandria Engineering Journal, 2024; vol. 101, pp. 52–61. doi: 10.1016/j.aej.2024.05.036.
[9] Saha S, Bera B, Das AK, Kumar N, Islam SH, and Park Y. Private blockchain envisioned access control system for securing Industrial IoT-based pervasive edge computing. IEEE Access, 2023; vol. 11, pp. 130206–130229. doi: 10.1109/ACCESS.2023.3333441.
[10] Zhao M, Shi C, and Yuan Y. Blockchain-based lightweight authentication mechanisms for Industrial Internet of Things and information systems. International Journal on Semantic Web and Information Systems, 2024; vol. 20, no. 1, pp. 1–30. doi: 10.4018/IJSWIS.334704.
[11] Dey P, Chaulya SK, and Kumar S. Hybrid CNN-LSTM and IoT-based coal mine hazards monitoring and prediction system. Process Safety and Environmental Protection, 2021; vol. 152, pp. 249–263. doi: 10.1016/j.psep.2021.06.005.
[12] Chen L. Recommendation for key derivation using pseudorandom functions. NIST Special Publication 800-108 Rev. 1, Update 1, National Institute of Standards and Technology, Feb. 2024. doi: 10.6028/NIST.SP.800-108r1-upd1.
[13] Yang P. ShangMi (SM) cipher suites for TLS 1.3. RFC 8998, Mar. 2021. doi: 10.17487/RFC8998.
[14] Dworkin MJ. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST SP 800-38D, 2007. doi: 10.6028/NIST.SP.800-38D.
[15] Standardization Administration of China. Information security technology—SM4 block cipher algorithm. GB/T 32907-2016, 2016.
[16] National Institute of Standards and Technology. Secure Hash Standard (SHS). FIPS PUB 180-4 (Update 1), 2015. doi: 10.6028/NIST.FIPS.180-4.
[17] Xu J, Wang C, and Jia X. A survey of blockchain consensus protocols. ACM Computing Surveys, 2023; vol. 55, no. 13s, Art. no. 278, pp. 1–35. doi: 10.1145/3579845.
[18] Guo R, Guo Z, Lin Z, and Jiang W. A hierarchical Byzantine fault tolerance consensus protocol for the Internet of Things. High-Confidence Computing, 2024; vol. 4, no. 3, Art. no. 100196. doi: 10.1016/j.hcc.2023.100196.
[19] Liu S, Zhang R, Liu C, Xu C, and Wang J. An improved PBFT consensus algorithm based on grouping and credit grading. Scientific Reports, 2023; vol. 13, no. 1, Art. no. 13030. doi: 10.1038/s41598-023-28856-x.
[20] Castro M and Liskov BH. Practical Byzantine fault tolerance. Proceedings of OSDI 1999, pp. 173–186. USENIX Association, 1999.
[21] Sun X, Yu FR, Zhang P, Sun Z, Xie W, and Peng X. A survey on zero-knowledge proof in blockchain. IEEE Network, 2021; vol. 35, no. 4, pp. 198–205. doi: 10.1109/MNET.011.2000473.
[22] Near J, Darais D, Lefkovitz N, and Howarth G. Guidelines for evaluating differential privacy guarantees. NIST Special Publication 800-226, National Institute of Standards and Technology, Mar. 2025. doi: 10.6028/NIST.SP.800-226.
[23] Islam M, Rehmani MH, and Chen J. Differentially private enhanced permissioned blockchain for private data sharing in industrial IoT. Information Sciences, 2024; vol. 658, Art. no. 119997. doi: 10.1016/j.ins.2023.119997.
[24] Milani S, Garlisi D, Carugno C, Tedesco C, and Chatzigiannakis I. Edge2LoRa: Enabling edge computing on long-range wide-area Internet of Things. Internet of Things, 2024; vol. 27, Art. no. 101266. doi: 10.1016/j.iot.2024.101266.
[25] IEEE. IEEE Standard for Ethernet. IEEE Std 802.3-2022, 2022. doi: 10.1109/IEEESTD.2022.9844436.
[26] Luo S, Hu T, Han N, and Qian Y. A smart-contract-based policy-domain access control framework for distributed Industrial IoT. IEEE Internet of Things Journal, 2024; vol. 11, no. 7, pp. 11427–11443. doi: 10.1109/JIOT.2023.3329541.
[27] Belchior R, Vasconcelos A, Guerreiro S, and Correia M. A survey on blockchain interoperability: Past, present, and future trends. ACM Computing Surveys, 2022; vol. 54, no. 8, Art. no. 168, pp. 1–41. doi: 10.1145/3471140.
[28] Kotey SD, Tchao ET, Ahmed AR, Agbemenu AS, Nunoo-Mensah H, Sikora A, Welte D, and Keelson E. Blockchain interoperability: The state of heterogenous blockchain-to-blockchain communication. IET Communications, 2023; vol. 17, no. 8, pp. 891–914. doi: 10.1049/cmu2.12594.
[29] Tong W, Yang L, Li Z, Jin X, and Tan L. Enhancing security and flexibility in the Industrial Internet of Things: Blockchain-based data sharing and privacy protection. Sensors, 2024; vol. 24, no. 3, Art. no. 1035. doi: 10.3390/s24031035.
[30] Liu Z, Wang S, Duan S, Ren L, and Wei J. Dynamic data integrity auditing based on hierarchical Merkle hash tree in cloud storage. Electronics, 2023; vol. 12, no. 3, Art. no. 717. doi: 10.3390/electronics12030717.
[31] Merkle RC. A digital signature based on a conventional encryption function. Advances in Cryptology—CRYPTO 1987, LNCS 293, 1988; pp. 369–378. doi: 10.1007/3-540-48184-2_32.
[32] Yu X, Zhang K, Suo Z, Wang J, Wang W, and Zou B. An efficient authentication scheme syncretizing physical unclonable function and revocable biometrics in Industrial Internet of Things. Journal of King Saud University—Computer and Information Sciences, 2024; vol. 36, Art. no. 102166. doi: 10.1016/j.jksuci.2024.102166.
[33] Burrows M, Abadi M, and Needham R. A logic of authentication. ACM Transactions on Computer Systems, 1990; 8(1):18–36. doi: 10.1145/77648.77649.

[34] Krawczyk H, Bellare M, and Canetti R. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, 1997. doi: 10.17487/RFC2104.
[35] Kuang Y, Wu Q, Chen R, and Liu X. Blockchain based lightweight authentication scheme for internet of things using lattice encryption algorithm. Computer Standards & Interfaces, 2025; vol. 93, Art. no. 103981. doi: 10.1016/j.csi.2025.103981.
[36] Kim D, Park H, and Park Y. A secure and lightweight authentication and key agreement protocol for blockchain-assisted IoT collaboration environments. Electronics, 2026; vol. 15, no. 12, Art. no. 2714. doi: 10.3390/electronics15122714.