Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

On-Chain Anchoring-Based Information Security on Iot for Secure Smart-City Using Pekeccak and Gsbdostlru

Pullaiah Chowdary Vutla

pvutla9127@ucumberlands.edu

University of the Cumberlands

Basker Ganesan

Triveni Yenugu

Jawaharlal Nehru Technological University Hyderabad

Article

Keywords: Information Security, Blockchain Anchoring, Secure Smart-City, Gated Recurrent Unit (GRU), Secretary Bird Optimization (SBO), InterPlanetary File System (IPFS), and Authentication

Posted Date: November 12th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11029218/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

ON-CHAIN ANCHORING-BASED INFORMATION SECURITY ON IoT FOR SECURE SMART-CITY USING PEKeccak AND GSBDOSTLRU

Mr. Pullaiah Chowdary Vutla[1], Mr. Basker Ganesan[2], Miss. Triveni Yenugu[3]
[1]*Research Scholar, Department of Information Technology, University of the Cumberlands, Williamsburg, KY 40769 USA*
[1]Corresponding author Email ID: pvutla9127@ucumberlands.edu
[2]*Director of Software Engineering*, 3232 Tayloe CT, Herndon VA 20171
[2]Email ID: baskerganesan@gmail.com
[3]*Software Engineer, Computer Science and Engineering (CSE), Jawaharlal Nehru Technological University Hyderabad, Telangana 500085, India*
[3]Email ID: *yenugutriveni944@gmail.com*

Abstract: For Internet of Things (IoT) data integrity, a highly secure information security framework for a Smart City (SC) environment is essential. Nevertheless, gateway-level log falsification was overlooked in traditional works. Therefore, this paper proposes Permutation Entropy-Keccak (PEKeccak)-based on-chain anchoring. At first, the sensor device, edge, gateway, and Blockchain (BC) are initialized, and Polynomial Rolling Digital Signature Algorithm (PRDSA)-based signature creation is carried out. After that, by utilizing Feistel Function-based Post-Quantum Cryptography (FFPQC), the data is sensed and secured. Then, PRDSA-based device authentication, Gated Secretary Bird Dawson Optimization Suish Transfer Learning Recurrent Unit (GSBDOSTLRU)-based intrusion detection, Shamir's Trigonometric Integral Secret Sharing (STISS)-based split-key storage, and off-chain data storage are done at the edge. Thereafter, in the gateway, edge authentication and on-chain anchoring are performed. Subsequently, gateway authentication and anchoring storage are carried out in the BC. Eventually, in the application layer, the key is reconstructed and decrypted. Hence, the data is anchored with a 98.91% Integrity Verification Success Rate (IVSR).

Keywords: *Information Security, Blockchain Anchoring, Secure Smart-City, Gated Recurrent Unit (GRU), Secretary Bird Optimization (SBO), InterPlanetary File System (IPFS), and Authentication.*

1. INTRODUCTION

In the SC infrastructure, the IoT has become a backbone for a variety of applications like traffic management, surveillance, public health, and so on (Hanumantharaju et al., 2025) (Qazi et al., 2024). However, information security challenges are introduced by the large-scale deployment of IoT devices, thereby compromising critical decision-making (Mrabet & Sliti, 2025). Therefore, in the SC environment, a robust framework that guarantees data confidentiality and intrusion resilience is necessary (Aldawsari, 2025). BC technology that includes a distributed ledger has emerged as a trusted model in recent times (Aborokbah, 2024). For intrusion detection, Long Short-Term Memory (LSTM) (Hazman et al., 2024) and Recurrent Neural Network (RNN) were utilized in numerous conventional works (Almasabi et al., 2024). However, the new attack in the dynamic environment can't be predicted (Muniswamy & Rathi, 2025). Additionally, the SC gateways are vulnerable to information tampering (Singh et al., 2024). Along with that, the IoT

nodes can't adapt to heavy cryptographic learning (Pervez et al., 2023). Thus, this research concentrates on incorporating PEKeccak-based BC anchoring, PRDSA-based secure authentication, and GSBDOSTLRU-based intrusion identification.

1.1 Research Overview

Research Gap: The sensor logs were altered, deleted, and fabricated by gateway-level log falsification, which was not addressed in any prevailing work, thus leading to the loss of security in the SC IoT environment.

Problem Statement: The prevailing works have some limitations, which are given below,

- ✱ Conventional work (Padma & Ramaiah, 2024) depended on long-term keys, which could be harvested and decrypted by utilizing quantum adversaries.
- ✱ In the existing SC model (Bekkali et al., 2023), multi-layer authentication was lacking, causing the injection of fake sensor values.
- ✱ The prevailing works didn't generalize to the new threats, resulting in inaccurate intrusion detection.

Significance and Scope of the Study: By incorporating intrusion detection, on-chain anchoring, authentication, and key management, an SC IoT security framework is developed in this research work.

Research Aim and Objective: Establishing information security for the IoT SC environment is this research's key aim. The research contributions are rendered below,

- ✓ A lightweight PEKeccak-based on-chain anchoring is carried out to prevent gateway-level log falsification.
- ✓ FFPQC-based data security and STISS-based split-key storage are accomplished to enhance data confidentiality.
- ✓ PRDSA is utilized for multi-layer authentication across sensors, edge, gateway, and BC.
- ✓ In GSBDOSTLRU-based intrusion detection, Transfer Learning (TL) is included for analyzing new patterns.

The paper is organized as: The literature survey is explained in Section 2, the proposed methodology is detailed in Section 3, the performance evaluation is described in Section 4, and the paper is concluded with future scope in Section 5.

2. LITERATURE SURVEY

(Padma & Ramaiah, 2024) introduced BC-based privacy preservation for SCs. Here, using Chebyshev polynomials, the data was protected. Thereafter, the digital assets were transmitted via smart contracts and stored in the BC. Therefore, secure data sharing in the IoT SC environment was achieved. Conversely, quantum adversaries compromised the security keys.

(Bekkali et al., 2023) assessed a BC-centric architecture for cyber-secure SCs. The sensor data was gathered in the perception layer and passed to the data-processing layer. Subsequently, using the practical Byzantine Fault Tolerance (pBFT) protocol, the transactions were processed on the

BC. Therefore, the cybersecurity challenges were addressed. Yet, data tampering occurred due to the absence of authentication.

(Syed et al., 2024) employed secure group authentication for IoT. Initially, using Dickson's polynomial, the secret key was generated. After that, to ensure integrity, keys were stored in the BC. Then, for authentication, the smart contract of the BC and the generated key were used, thereby avoiding data tampering. Nevertheless, the intrusion in the network wasn't determined. Therefore, the transmission performance was reduced.

(Margret & Julie, 2024) incorporated the smart contract application for SCs using BC technology. Here, based on the Keccak-based hash values, the registration in the BC was carried out. Further, access control was maintained for transmitting the data. Therefore, the file management was securely performed. In contrast, the trust of multiple users was affected by the log falsification.

(Ahmad et al., 2024) developed IoT and BC-centric transparent supply chain management in SCs. The service controller and an IoT controller were adopted, where local storage was handled. Afterward, to manage the data access request, the Revised Fitness-based Political Optimizer (RF-PO) queue model was utilized. Hence, the data feasibility was maintained. However, the data was not secured, thus losing trust in accessing the data.

3. PROPOSED METHODOLOGY REGARDING INFORMATION SECURITY ON IoT FOR SECURE SMART CITY

Figure 1 depicts the PEKeccak-based on-chain gateway anchoring and GSBDOSTLRU-based intrusion detection for a secure SC framework.

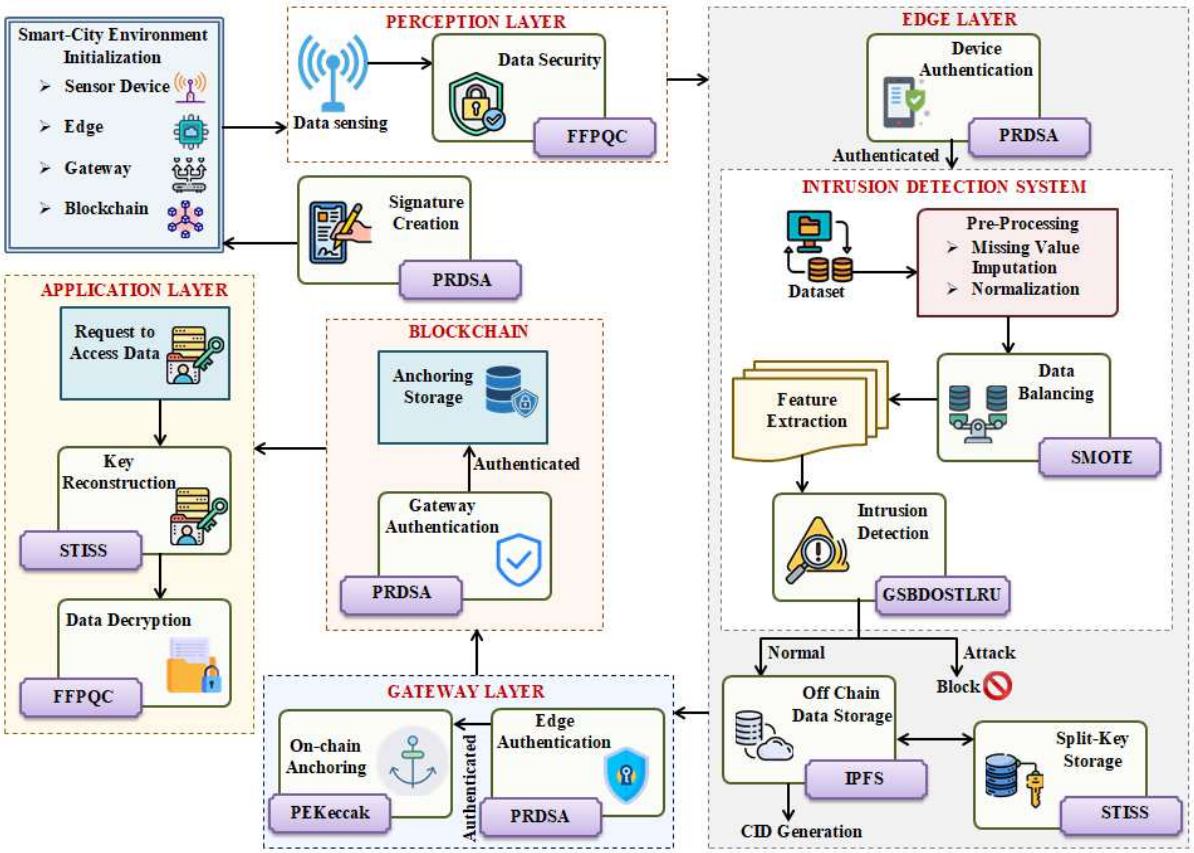


Figure 1: Structure of the Proposed System

3.1 Smart-City Environment Initialization

The proposed system begins with the initialization of sensors (K), edge (P), gateway (D), and BC (M) of the SC environment. At the time of initialization, the signature is created, which is expressed as follows,

Signature Creation:

Here, using PRDSA, the signatures required for the authentication of (K), (P), (D), and (M) are created. For signature creation, the Digital Signature Algorithm (DSA) is chosen, which efficiently generates signatures. However, the hash function (δ) in DSA can be hacked. Thus, for hash function creation, the Polynomial Rolling (PR) function is used, which utilizes large mod primes.

$$\delta = \left[\sum a \times C\right] \bmod N \tag{1}$$

Where, the integer encoder, the Identifier (ID) of each layer, and a large prime modulus are signified as (a, C, N), respectively. Further, the signature $(\Im)$ is generated as,

$$\Im = (K, P, D, M)C + \delta \tag{2}$$

Now, $(\Im)$ is stored in BC and utilized for authentication. Data is further sensed.

3.2 Perception Layer

The perception (sensor) layer is responsible for collecting raw data (H) and securing it.

Data Sensing and Security:

Initially, from $(K), (H)$ is sensed and secured using FFPQC. Here, for data security, Post-Quantum Cryptography (PQC) is selected, which varies the quantum states. Nevertheless, the random secret key might be attacked. Therefore, for secret key (ε) (quantum key) creation, the Feistel Function (FF), which repeats the substitution and permutation, is utilized.

$$\varepsilon = c \oplus [\phi, d] \tag{3}$$

Here, the random value, round sub-key, and Feistel round function are represented as (c, ϕ, d), respectively, and the Exclusive OR operator is indicated as $(\oplus)$. Subsequently, the eavesdropping is checked, and (H) is encrypted.

$$H^* = H \times (\gamma, e) + \varepsilon \tag{4}$$

Here, the secured data, photons, and polarization factor are notated as (H^*, γ, e), correspondingly. Then, (H^*) is passed to the edge layer.

3.3 Edge Layer

The edge (P) is used for local processing and storage of (H^*).

3.3.1 Device Authentication

At first, (P) authenticates the device (K) using PRDSA. In Section 3.1, $(\Im)$ is generated, which is used for authentication (A).

$$A = \begin{cases} A'' & \forall[\Im(K) = \Im(M)] \\ \dddot{A} & \forall[\Im(K) \neq \Im(M)] \end{cases} \tag{5}$$

Here, (P) checks $(\Im)$, which is stored in the BC (M). If $(\Im)$ send by (K) is equal to $(\Im)$ in (M), then the device (K) is authenticated (A''); otherwise, the device (K) is unauthenticated $(\dddot{A})$. The intrusion is detected once (A'') is attained.

3.3.2 Intrusion Detection System

The Intrusion Detection System (IDS) is trained to detect intrusions during the transmission of (H^*).

Dataset: Initially, the IDS dataset is gathered. The network data (F) present in the dataset is signified as,

$$F = \left[F^1, F^2, F^3, ..., F^n\right] \tag{6}$$

Where, the (n^{th}) network data is indicated as (F^n). Further, pre-processing is performed.

Pre-Processing: For pre-processing (F), the missing values in (F) are initially imputed using the mean of the neighboring values. The Missing Value Imputation (MVI) output is represented as $(\hat{F})$. Afterward, using min-max normalization, $(\hat{F})$ is normalized in the range of 0 to 1.

$$\widetilde{F} = \frac{\hat{F} - \hat{F}^{\min}}{\hat{F}^{\max} - \hat{F}^{\min}} \tag{7}$$

Here, the normalized (pre-processed) data, minimum $(\hat{F})$, and maximum $(\hat{F})$ are illustrated as $(\widetilde{F}, \hat{F}^{\min}, \hat{F}^{\max})$, respectively. Next, $(\widetilde{F})$ is balanced.

Data Balancing: Here, utilizing the Synthetic Minority Over-sampling Technique (SMOTE) that effectively generates realistic data points, $(\widetilde{F})$ is balanced. Primarily, for the minority class $(\widetilde{F}_2)$, the synthetic data (k) is generated regarding the majority class $(\widetilde{F}_1)$. The balanced data $(\ddot{\dot{G}})$ is articulated as,

$$\ddot{\dot{G}} = \left(k, \widetilde{F}_2\right) + \widetilde{F}_1 \tag{8}$$

Then, features are extracted.

Feature Extraction: From $(\ddot{\dot{G}})$, the features (G), such as protocol, traffic volume characteristics, number of packets transmitted, session states, activity time, flow duration, packet timing, hop counts, and so on, are extracted. After that, intrusion is detected.

Intrusion Detection: GSBDOSTLRU is utilized for intrusion detection. Here, for intrusion identification, the GRU that handles temporal dependencies is selected. Yet, GRU has slow convergence and low learning efficiency. Hence, the hyperparameters are tuned using SBO, which uses the hunting and escaping strategy, to avoid a low learning rate. Nevertheless, SBO has a local minima issue; thus, the Dawson Function (DF) is utilized. Also, the Suish activation function and TL are used to mitigate slow convergence and to learn new patterns, respectively. Figure 2 represents the architecture of GSBDOSTLRU.

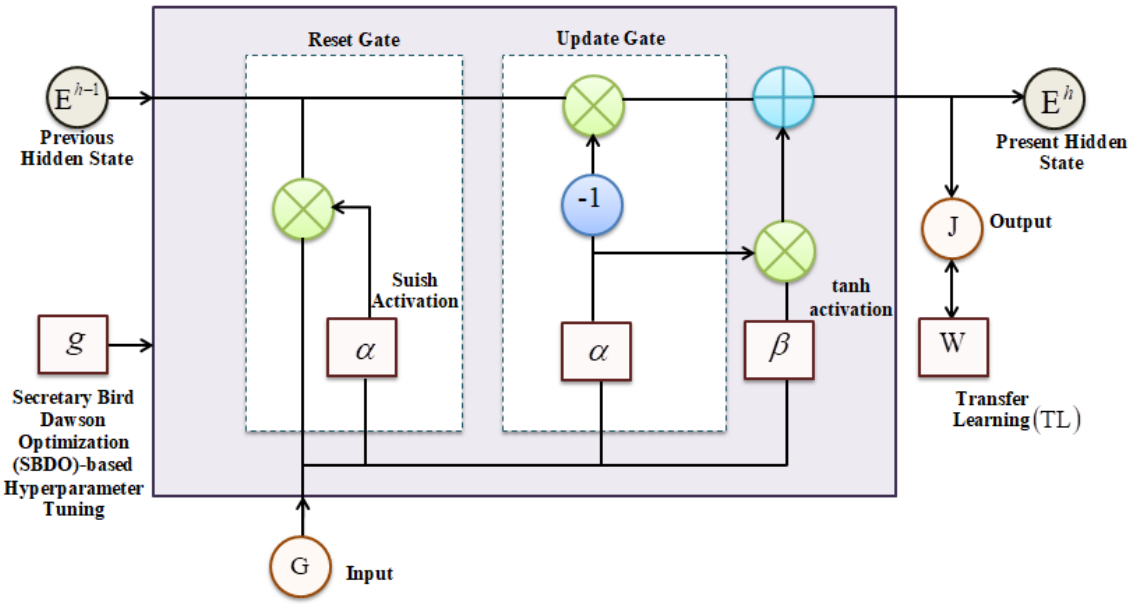


Figure 2: Proposed GSBDOSTLRU Classifier

Firstly, using Secretary Bird Dawson Optimization (SBDO), the hyperparameters are tuned. The initial position of the secretary bird (hyperparameters) is notated as (T). Based on maximum classification accuracy (η), the fitness $(\Re)$ to choose the optimal solution is set.

$$\Re = \max(\eta) \tag{9}$$

Subsequently, based on exploration (hunting of snakes) and exploitation (escaping from predators) of the secretary bird, the position (T) is updated. The new positions (T^1, T^2) regarding exploration and exploitation are articulated as,

$$T^1 = T + (y \times T) \tag{10}$$

$$T^2 = T^1 + \left[(2y-1) \times \left(1 - \frac{1}{i} \right) \times T^1 \right] \tag{11}$$

$$y = \exp^{-T^2} \int \exp^{T^2} \partial T \tag{12}$$

Where, the DF, which enables quasi-oscillatory behavior and avoids the local minima, and the present iteration are given as (y, i), respectively. Therefore, (η) is applied to (T^1, T^2) and the optimal hyperparameter (g) is tuned. Thereafter, (G) is fed into GSBDOSTLRU. The information to be added or deleted is decided by the reset gate. In addition, by utilizing the update gate, the memory is updated. The reset gate and update gate outputs (U_1, U_2) are rendered as,

$$U_1 = \langle [(G \times w) + v] \alpha \rangle \otimes E^{h-1} \tag{13}$$

$$U_2 = \langle 1 - [(G \times w) + v]\alpha \rangle \otimes U_1 \tag{14}$$

$$\alpha = \max[G, (G \times \exp|G|)] \tag{15}$$

Here, the weight, bias, Suish activation that enables the strongest gradient flow, and the previous hidden state with time (h) are illustrated as (w, v, α, E^{h-1}), correspondingly. At last, the intrusion detection (J) is conducted as,

$$J = [(G \times w) + v]\beta \oplus U_2 \quad \forall [J = (J^m, J^b)] \tag{16}$$

$$\beta = \frac{\exp^G - \exp^{-G}}{\exp^G + \exp^{-G}} \tag{17}$$

Here, the tanh activation function, normal class, and attack classes, such as Cross-Site Scripting (XSS), Denial of Service (DoS), scanning, Distributed DoS (DDoS), injection, password, backdoor, ransomware, and Man-In-The-Middle (MITM), are denoted as (β, J^m, J^b), respectively. Lastly, TL (W) is applied to learn the new patterns.

Pseudo-code for GSBDOSTLRU

Input: Extracted Feature (G)

Output: Intrusion Detection (J)

Begin

Tune hyperparameters (g) using SBDO

Initialize (w, v)

While (G)

Evaluate activation

$$\alpha = \max[G, (G \times \exp|G|)], \ \beta = \frac{\exp^G - \exp^{-G}}{\exp^G + \exp^{-G}}$$

Determine reset gate

$$U_1 = \langle [(G \times w) + v]\alpha \rangle \otimes E^{h-1}$$

Update memory

$$U_2 = \langle 1 - [(G \times w) + v]\alpha \rangle \otimes U_1$$

Equate output

$$J = [(G \times w) + v]\beta \oplus U_2 \quad \forall [J = (J^m, J^b)]$$

End while

Obtain (K)

Apply TL (W)

End

The intrusion is detected after training the IDS. If (J^b) is attained, then the transmission of (H^*) is blocked. If (J^m) is detected, then transmission is continued.

3.3.3 Split-Key Storage & Off-Chain Storage

Now, by utilizing STISS, the secret key (ε) generated using FFPQC (Section 3.2) is split. Here, for split-key storage, Shamir's Secret Sharing (SSS), which splits and stores the keys in different locations, is used. However, the scheme is broken by weak randomness in share generation. Thus, in SSS, the Trigonometric Integral (TI) (λ) is used, which integrates the share freshness using pseudo-random transformation masks.

$$\lambda = \int [\sin(q)/q] \partial q \tag{18}$$

Where, the number of shares needed to reconstruct the key is exemplified as (q). Subsequently, concerning (λ) and random coefficient $(B), (\varepsilon)$ is split.

$$\vec{\varepsilon} = \left\{\varepsilon^1, \varepsilon^2, ..., \varepsilon^{q-1}, \varepsilon^q\right\} \times (\lambda, B) \tag{19}$$

Here, the split-key is represented as $(\vec{\varepsilon})$, and the split-key of the (q^{th}) share is indicated as (ε^q). Next, $(\vec{\varepsilon})$ and (H^*) are stored in the IPFS off-chain storage that retrieves the data (H^*) regarding the Content Identifier (CID) $(\aleph)$.

$$\aleph = \left[H^*, \vec{\varepsilon}, L\right] \tag{20}$$

Where, the storage location of each data is depicted as (L). Further, $(\aleph)$ is given to the gateway layer.

3.4 Gateway Layer

The gateway layer serves as a bridge for data transmission. Here, edge authentication and on-chain anchoring are carried out.

3.4.1 Edge Authentication

Here, using PRDSA, the edge (P) is authenticated by the gateway (D). After successful authentication, the on-chain anchoring continues in the gateway.

3.4.2 On-Chain Anchoring

By using PEKeccak, the hash is generated for $(\aleph)$ to perform on-chain anchoring. Here, for on-chain anchoring, the Keccak model, which generates a unique hash and ensures security, is chosen. Yet, more memory is required by the internal state size of Keccak. Thus, for internal Keccak state (S) estimation, the Permutation Entropy (PE), which captures the order relations between values in a time series, is applied.

$$S = -\sum_{u} \mu(Q) \log[\mu(Q)] \tag{21}$$

Where, the ordinal patterns of $(\aleph)$, the number of (Q), and the probability of (Q) are specified as (Q,u,μ), correspondingly. Now, to transform the input $(\aleph)$, a lightweight mask (Z) is generated.

$$Z = \aleph * S \tag{22}$$

At last, a hash (V) is generated for $(\aleph)$ and anchored on-chain in the Merkle root of the BC (M).

$$V = \varsigma + (Z, \aleph) \tag{23}$$

Here, the sponge function is denoted as (ς). Thus, the gateway anchoring turns out to be faster and aids in ensuring information security in an SC IoT environment.

Pseudo-code for PEKeccak

Input: CID $(\aleph)$
Output: Hash (V)

Begin
Initialize (Q,u,μ)
For $(\aleph)$
Determine PE-based states
$S = -\sum_{u} \mu(Q)\log[\mu(Q)]$
Transform input to mask $Z = \aleph * S$
Generate hash $V = \varsigma + (Z, \aleph)$
Anchor (V) on-chain
End for
Return Hash (V)
Pass (V) to Blockchain (M)
End

Further, in BC, gateway authentication is carried out.

3.5 Blockchain

Immutable logging and secured anchoring of (V) are offered by BC.

Gateway Authentication: Initially, using PRDSA, the gateway (D) is authenticated by BC (M). Once the authentication is successful, (V) from the gateway is stored in (M).

Anchoring Storage: Here, (V) is stored in the Merkle root of (M), thereby ensuring tamper-evident storage and traceability. Lastly, the application layer provides the service.

3.6 Application Layer

The Merkle proofing, key reconstruction, and data decryption are carried out in the application layer.

Request to Access Data: Initially, the trusted application user in the IoT SC requests to access the data.

Key Reconstruction: $(V)$is verified as per the request to attain the CID$(\aleph)$. Once the CID is attained, the split keys$(\vec{\varepsilon})$are reconstructed by utilizing STISS. Thus, the secret key$(\varepsilon)$is acquired. Further, the secured data is decrypted.

Data Decryption: Once $(\varepsilon)$is attained, the CID is utilized for identifying the location of the data in the IPFS storage. Next, data decryption is carried out.

$$H = H^{*}(\aleph) - \varepsilon \tag{24}$$

Hence, the sensed data$(H)$is securely sent to the user. Therefore, the information in the IoT SC environment is secured by the proposed work. The proposed models' performance analysis is explained as follows,

4. RESULTS AND DISCUSSION

Here, the proposed models' performance evaluation is done. The proposed model is employed using the PYTHON tool.

4.1 Dataset Description

By utilizing the "NetFlow Telemetry Of Network IOT Version 2 (NF-TON-IOT-V2)" dataset, IDS is trained. This dataset consists of network flow-based features that describe the characteristics of network traffic. The dataset link is given in Table 1 and in the reference section. By using SMOTE, the data is balanced. Table 1 provides the dataset details.

Table 1: Dataset Details

Dataset Name	NF-TON-IOT-V2	
Dataset Link	https://www.kaggle.com/datasets/shahidabbas76/nf-ton-iot-v2-full-dataset	
Class	**Original Data**	**Balanced Data**
Normal	6099469	6099469
Scanning	3781419	6099469
XSS	2455020	6099469
DDoS	2026234	6099469
Password	1153323	6099469

Injection	684465	6099469
DoS	712609	6099469
Backdoor	16809	6099469
MITM	7723	6099469
Ransomware	3425	6099469

After balancing the data from each class, 80% (4879575) and 20% (1219894) of the data are utilized for training and testing the IDS, respectively.

4.2 Performance Validation

In this section, the proposed models' performance is validated and contrasted with the conventional methods.

Table 2: Comparative Analysis of PEKeccak

Methods	IVSR (%)	CRL (%)	Anchoring Latency (ms)
Proposed PEKeccak	98.91	99.01	1253
Keccak	96.21	95.87	3821
BLAKE	93.43	92.96	5782
HMAC	91.18	90.53	7492
SHA	89.27	88.74	9841

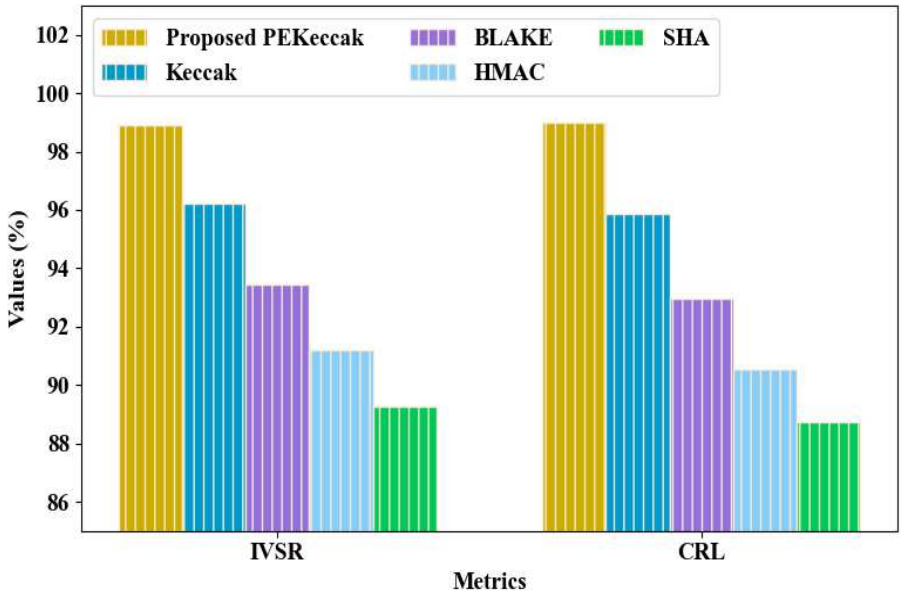


Figure 3: Graphical Comparison of PEKeccak

According to Figure 3 and Table 2, the hash was anchored by the proposed PEKeccak with an IVSR of 98.91%, a Collision Resistance Level (CRL) of 99.01%, and an anchoring latency of 1253ms. However, when contrasted with the proposed model, the prevailing Keccak, BLAKE model, Hash-centric Message Authentication Code (HMAC), and Secure Hash Algorithm (SHA) attained lower IVSR, lower CRL, and higher anchoring latency. Thus, the proposed method's performance was enhanced as a result of the utilization of PE.

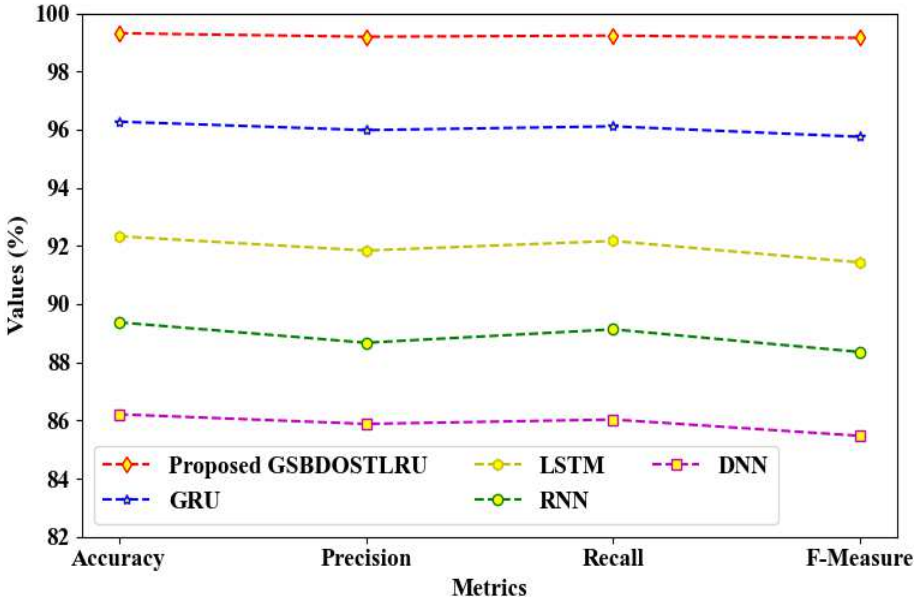


Figure 4: Comparative Analysis of GSBDOSTLRU

By using SBDO, the proposed classifier tuned the hyperparameters. Also, for effective processing, the proposed classifier utilized TL and Suish activation. Therefore, as exhibited in Figure 4, the intrusion was detected with 99.31% accuracy, 99.19% precision, 99.23% recall, and 99.15% F-Measure. Nevertheless, the conventional GRU, LSTM, RNN, and Deep Neural Network (DNN) obtained an average accuracy of 91.05%, precision of 90.59%, recall of 90.86%, and F-Measure of 90.25%. Thus, when contrasted with the conventional classifiers, the proposed GSBDOSTLRU performed better.

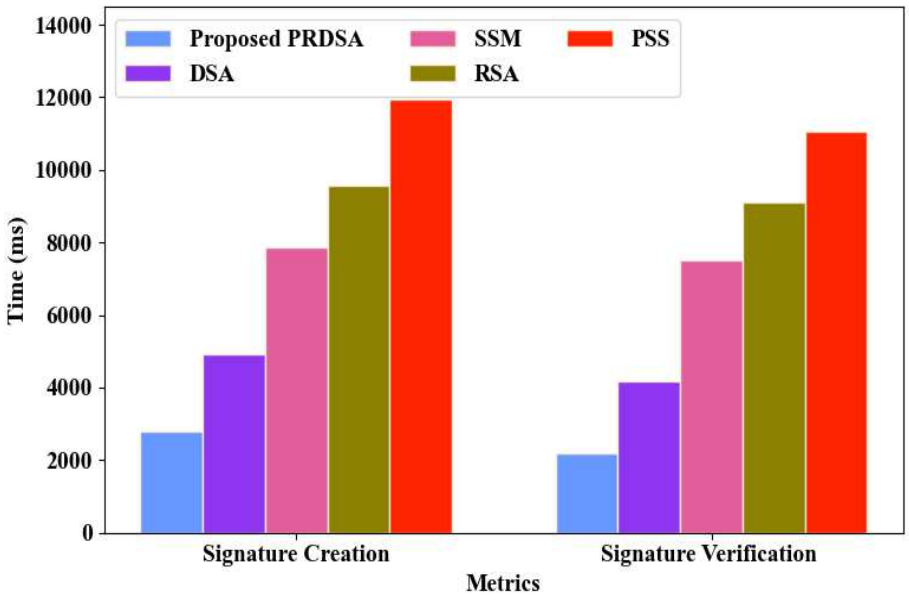


Figure 5: Graphical Comparison of PRDSA

As illustrated in Figure 5, the multi-layers in the IoT SC environment were authenticated by the proposed model with a Signature Creation Time (SCT) and Signature Verification Time (SVT) of 2784ms and 2191ms, correspondingly. Nevertheless, the conventional DSA, Schnorr Signature Model (SSM), Rivest-Shamir-Adleman (RSA), and Probabilistic Signature Scheme (PSS) attained SCTs of 4928ms, 7854ms, 9572ms, and 11943ms, and SVTs of 4178ms, 7523ms, 9105ms, and 11073ms, respectively. Hence, the proposed authentication process was improved due to the generation of a hash value using PR.

Table 3: Comparative Analysis of FFPQC

Techniques	Encryption Time (ms)	Decryption Time (ms)	Security Level (%)	Attack Level (%)
Proposed FFPQC	2294	2110	98.47	1.27
PQC	3984	3728	95.13	3.84
ECC	5193	4982	92.27	5.98
AES	6278	6021	89.28	8.49
DES	8137	8042	87.63	9.83

Using FF, the secret key was generated in the proposed model. Therefore, as per Table 3, the proposed FFPQC secured the sensor data with an encryption time, decryption time, security level, and attack level of 2294ms, 2110ms, 98.47%, and 1.27%, respectively. On the other hand, Elliptic Curve Cryptography (ECC), traditional PQC, Advanced Encryption Standard (AES), and Data Encryption Standard (DES) had higher encryption and decryption times, lower security

levels, and higher attack levels. Therefore, the data was effectively secured by the proposed FFPQC.

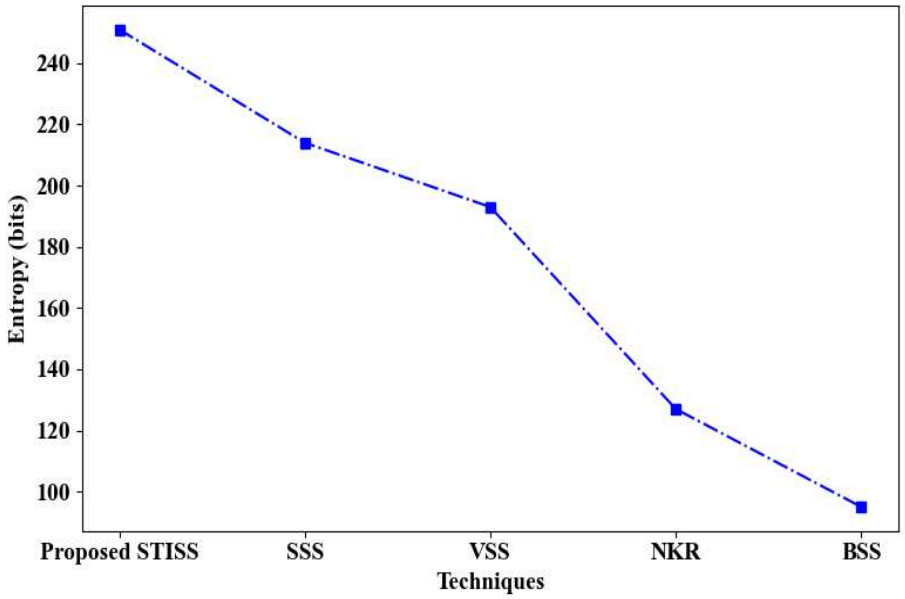


Figure 6: Comparative Analysis of STISS

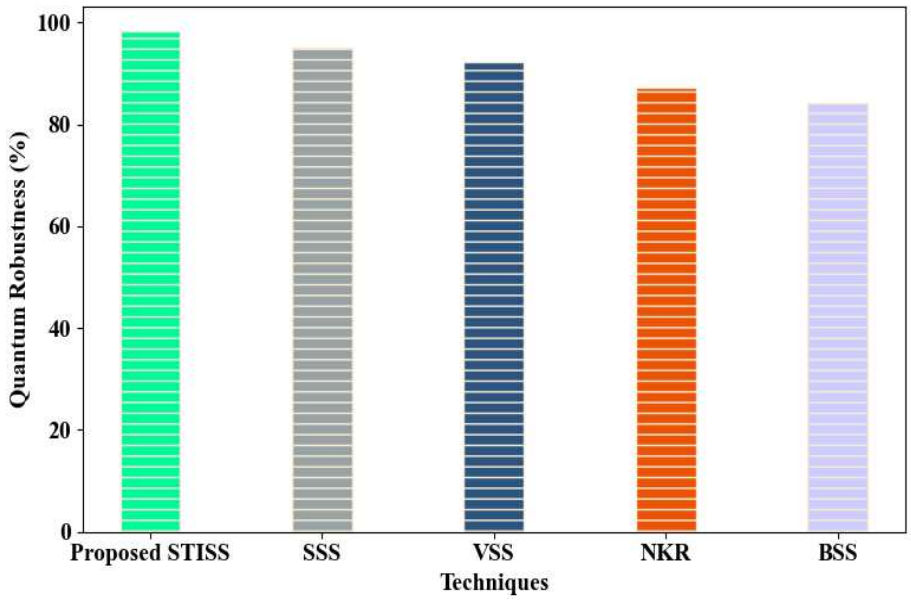


Figure 7: Comparison regarding Quantum Robustness

As demonstrated in Figures 6 and 7, the keys were split by the proposed STISS with 251bits of entropy and 98.21% quantum robustness. Nevertheless, the conventional SSS, Verifiable Secret Sharing (VSS), Naïve Key Replication (NKR), and Bitwise Split Storage (BSS) achieved entropy of 214bits, 193bits, 127bits, and 95bits, and quantum robustness of 95.03%, 92.18%,

87.17%, and 84.32%, correspondingly. Therefore, in the proposed model, the key splitting was improved by the TI-based share integration.

Table 4: Existing Works Comparison

Study	Methods	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
Proposed Work	GSBDOSTLRU	99.31	99.19	99.23	99.15
(Kumar et al., 2025)	AIAD	98	-	-	-
(Ali et al., 2024)	ACLR	96.98	96.91	96.93	96.92
(Sathyabama & Katiravan, 2025)	DNN	99.18	-	-	-
(Mancy & Naith, 2025)	DSVDD-CAE	93	94	92	93
(Malik et al., 2023)	DRL	98.765	98.765	97.442	96.115

The comparison of the proposed and the prevailing works concerning intrusion detection is exhibited in Table 4. For intrusion detection, the existing works used Artificial Intelligence Anomaly Detector (AIAD), Artificial-Neural-Network Convolutional-Neural-Network LSTM RNN (ACLR), Deep Support Vector Data Description Contractive AutoEncoder (DSVDD-CAE), and Deep Reinforcement Learning (DRL). The prevailing work either failed to analyze the new patterns or didn't balance the data before processing or attained slow convergence. Therefore, the prevailing works attained lower accuracy, precision, recall, and F-Measure than the proposed work. Hence, the intrusion was effectively detected by the proposed system.

5. CONCLUSION

The IoT SC environment with sensor data was secured in this paper. The IoT SC environment was initialized. After that, using PRDSA, the signature was created within 2784ms. Then, the sensed data was secured by utilizing FFPQC with a 98.47% security level. Subsequently, the device was authenticated. Also, using GSBDOSTLRU, the intrusion was detected in the edge layer, which attained an accuracy of 99.31%. The keys were split by utilizing STISS with a quantum robustness of 98.21%, followed by off-chain storage. Thereafter, the edge was authenticated. After that, using PEKeccak, the on-chain gateway anchoring was performed with an IVSR of 98.91%. Next, gateway authentication and anchoring storage were carried out in BC. Eventually, the keys were securely reconstructed, and the data was decrypted in the application layer within 2110ms. Thus, the IoT SC environment was secured by the proposed framework.

Practical Implication: This model is designed to operate across heterogeneous IoT devices for city-scale deployment.

Future Scope: To further enhance the network's security, future work will focus on the detection of synchronized attacks on the digital infrastructure and IoT devices.

Declarations

Conflict of interest: The authors declare that they have no conflict of interest.

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

Consent of publication: Not applicable.

Availability of data and materials: The dataset used in this study is described in detail in the Dataset Description section, and its publicly accessible source is cited in the References section and in Table 1. Additional data, experimental results, and other materials generated and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Competing interests: The authors declare that they have no competing interests

Funding Declaration: This work has no funding resources.

Author's Contributions: Pullaiah Chowdary Vutla: Writing– original draft, Visualization, Supervision, Investigation, Conceptualization. Basker Ganesan: Writing– review & editing, Validation, Supervision, Formal analysis. Triveni Yenugu: Writing– review & editing, Resources, Methodology.

Acknowledgement: We thank the anonymous referees for their useful suggestions.

REFERENCES:

Dataset Link: https://www.kaggle.com/datasets/shahidabbas76/nf-ton-iot-v2-full-dataset

Aborokbah, M. M. (2024). A Novel Intrusion Detection Model for Enhancing Security in Smart City. *IEEE Access*, *12*, 107431–107444. https://doi.org/10.1109/ACCESS.2024.3438619

Ahmad, A. Y. A. B., Verma, N., Sarhan, N. M., Awwad, E. M., Arora, A., & Nyangaresi, V. O. (2024). An IoT and Blockchain-Based Secure and Transparent Supply Chain Management Framework in Smart Cities Using Optimal Queue Model. *IEEE Access*, *12*, 51752–51771. https://doi.org/10.1109/ACCESS.2024.3376605

Aldawsari, H. (2025). A blockchain-based approach for secure energy-efficient IoT-based Wireless Sensor Networks for smart cities. *Alexandria Engineering Journal*, *126*, 1–7. https://doi.org/10.1016/j.aej.2025.04.052

Ali, M., Shahroz, M., Mushtaq, M. F., Alfarhood, S., Safran, M., & Ashraf, I. (2024). Hybrid

Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment. *IEEE Access*, *12*, 40682–40699. https://doi.org/10.1109/ACCESS.2024.3376400

Almasabi, A. M., Khemakhem, M., Eassa, F. E., Ahmed Abi Sen, A., Alkhodre, A. B., & Harbaoui, A. (2024). A Smart Framework to Detect Threats and Protect Data of IoT Based on Machine Learning. *IEEE Access*, *12*, 176833–176844. https://doi.org/10.1109/ACCESS.2024.3498603

Bekkali, A. El, Essaaidi, M., & Boulmalf, M. (2023). A Blockchain-Based Architecture and Framework for Cybersecure Smart Cities. *IEEE Access*, *11*, 76359–76370. https://doi.org/10.1109/ACCESS.2023.3296482

Hanumantharaju, R., Shreenath, K. N., Sowmya, B. J., Supreeth, S., Dayananda, P., Shivashankar, R., & Shruthi, G. (2025). Consortium blockchain-based IoT identity management for smart systems. *Cogent Engineering*, *12*(1), 1–34. https://doi.org/10.1080/23311916.2025.2573142

Hazman, C., Guezzaz, A., Benkirane, S., & Azrour, M. (2024). Enhanced IDS with Deep Learning for IoT-Based Smart Cities Security. *Tsinghua Science and Technology*, *29*(4), 929–947. https://doi.org/10.26599/TST.2023.9010033

Kumar, A., Sharma, B., & Noonia, A. (2025). Secure blockchain based intrusion detection for IoT networks. *Discover Computing*, *28*(1), 1–29. https://doi.org/10.1007/s10791-025-09754-4

Malik, V., Mittal, R., Mavaluru, D., Narapureddy, B. R., Goyal, S. B., Martin, R. J., Srinivasan, K., & Mittal, A. (2023). Building a Secure Platform for Digital Governance Interoperability and Data Exchange Using Blockchain and Deep Learning-Based Frameworks. *IEEE Access*, *11*, 70110–70131. https://doi.org/10.1109/ACCESS.2023.3293529

Mancy, H., & Naith, Q. H. (2025). SwinIoT: A Hierarchical Transformer-Based Framework for Behavioral Anomaly Detection in IoT-Driven Smart Cities. *IEEE Access*, *13*, 48758–48774. https://doi.org/10.1109/ACCESS.2025.3551207

Margret, M. K., & Julie, E. G. (2024). Smarter and resilient smart contracts applications for smart cities environment using blockchain technology. *Automatika Journal for Control, Measurement, Electronics, Computing and Communications*, *65*(2), 572–583. https://doi.org/10.1080/00051144.2024.2307228

Mrabet, M., & Sliti, M. (2025). Towards Secure, Trustworthy and Sustainable Edge Computing for Smart Cities: Innovative Strategies and Future Prospects. *IEEE Access*, *13*, 174236–174253. https://doi.org/10.1109/ACCESS.2025.3602390

Muniswamy, A., & Rathi, R. (2025). Trust-Based Consensus and ABAC for Blockchain Using Deep Learning to Secure Internet of Things. *Applied Artificial Intelligence An International Journal*, *39*(1), 1–48. https://doi.org/10.1080/08839514.2025.2459461

Padma, A., & Ramaiah, M. (2024). Blockchain Based an Efficient and Secure Privacy Preserved Framework for Smart Cities. *IEEE Access*, *12*, 21985–22002.

https://doi.org/10.1109/ACCESS.2024.3364078

Pervez, Z., Khan, Z., Ghafoor, A., & Soomro, K. (2023). SIGNED: Smart cIty diGital twiN vErifiable Data Framework. *IEEE Access*, *11*, 29430–29446. https://doi.org/10.1109/ACCESS.2023.3260621

Qazi, E.-U.-H., Zia, T., Faheem, M. H., Shahzad, K., Imran, M., & Ahmed, Z. (2024). Zero-Touch Network Security (ZTNS): A Network Intrusion Detection System Based on Deep Learning. *IEEE Access*, *12*, 141625–141638. https://doi.org/10.1109/ACCESS.2024.3466470

Sathyabama, A. R., & Katiravan, J. (2025). Enhancing anomaly detection and prevention in Internet of Things (IoT) using deep neural networks and blockchain based cyber security. *Scientific Reports*, *15*(1), 1–20. https://doi.org/10.1038/s41598-025-04164-4

Singh, N. J., Hoque, N., Singh, K. R., & Bhattacharyya, D. K. (2024). Botnet-based IoT network traffic analysis using deep learning. *Security and Privacy*, *7*(2), 1–41. https://doi.org/10.1002/spy2.355

Syed, S. A., Manickam, S., Uddin, M., Alsufyani, H., Shorfuzzaman, M., Selvarajan, S., & Mohammed, G. B. (2024). Dickson polynomial-based secure group authentication scheme for Internet of Things. *Scientific Reports*, *14*(1), 1–20. https://doi.org/10.1038/s41598-024-55044-2