Computer Modeling in Engineering & Sciences

Tech Science Press

https://doi.org/10.32604/cmes.2026.077316

ARTICLE

Constructing a Dynamic Trust Assessment Mechanism Combining Zero Knowledge Proof with Unsupervised Learning

Nai-Wei Lo[1], Cheng-I Lin[2], Chih-Chieh Chang[3,*], Chi-Yang Chang[4] and Tran Thi Luu Ly[1]

[1]Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan
[2]Graduate Institute of Management, National Taiwan University of Science and Technology, Taipei, Taiwan
[3]School of Management, National Taiwan University of Science and Technology, Taipei, Taiwan
[4]Graduate Institute of A.I. Cross-Disciplinary Technology, National Taiwan University of Science and Technology, Taipei, Taiwan

*Corresponding Author: Chih-Chieh Chang. Email: ccchang@mail.ntust.edu.tw

Received: 06 December 2025; Accepted: 24 February 2026; Published: 27 April 2026

ABSTRACT: The growing frequency of malicious attacks on Internet of Things (IoT) devices has rendered conventional approaches with static label-dependent risk assessment models obsolete, especially when coping with unknown and continuously evolving threats. To mitigate these challenges, a novel dynamic trust evaluation framework approach is proposed in this work. The proposed framework utilized unsupervised learning and zero-knowledge proofs to assess device risks in complex environments adaptively, with an accuracy rate of 98.96% for normal clustering and 95.39% for anomalies. K-means clustering algorithm is leveraged to distinguish risk patterns with an additional Decision Tree classification algorithm to analyze the distinguishing characteristics of the behaviors of normal and anomalous devices. The architecture is evaluated in a simulated environment based on real device interaction, with various malicious attacks proportions. In addition, Zero Trust Architecture is integrated into this novel framework to ensure no implicit trust exists between devices, which enforces trust assessment before any collaboration or data exchange.

KEYWORDS: Zero knowledge proof; zero trust architecture; decision tree; trust assessment; unsupervised learning

1 Introduction

The Internet of Things (IoT) has been deeply integrated into our daily lives, spanning from home automation systems to smart manufacturing, thanks to the rapid advancement of technology. The intelligent connectivity and data exchange capabilities of these devices have significantly enhanced the convenience and efficiency of both personal and industrial applications. However, the widespread deployment of IoT devices has also introduced a corresponding increase in security risks, as the expanding range and functionality of these devices present new opportunities for cyberattacks. These smart devices, which process vast amounts of data, are becoming prime targets for hackers, posing substantial threats to both individual and organizational security [1–4].

In traditional network environments, once devices have completed initial communication and trust assessments, they typically gain access to other devices within the network. While this design promotes ease of use, it also increases the system's vulnerability. For instance, an attacker could compromise a smart home device, such as a light bulb or assistant, and use it to infiltrate other devices within the network or form a botnet for launching distributed denial-of-service (DDoS) attacks, causing significant damage. Current security frameworks, which rely on static rules and predefined policies, struggle to address the

increasingly sophisticated and varied attack methods. As the number of IoT devices grows and network structures become more complex, attackers can exploit a broader range of attack vectors and advanced techniques to compromise system security. While traditional defense mechanisms, such as firewalls and intrusion detection systems, are still useful, they are insufficient for fully countering the evolving security threats in modern network environments [4–6]. Consequently, there is an urgent need for new security architectures to protect the integrity and security of IoT systems.

To mitigate these escalating and dynamic security threats, the National Institute of Standards and Technology (NIST) has proposed the adoption of Zero Trust Architecture (ZTA) [7]. ZTA operates on the principle of "never trust, always verify," requiring strict security checks and controls for every device and request, regardless of the device's location or network segment. This approach provides a robust defense against the complex and varied attack patterns in IoT environments, enhancing the security of both device collaboration and data exchange. Under ZTA, each interaction between devices necessitates a trust assessment, demanding the development of rules for continuous evaluation of security threats to devices and users.

In ZTA-driven IoT environments, credit (trust) evaluation can be formulated as mapping heterogeneous behavioral evidence, such as traffic statistics, authentication outcomes, policy compliance signals, and interaction histories, into a dynamic trust/risk score that governs access decisions [8]. In practice, such scores are commonly used to enforce *trust thresholds* prior to participation; for example, an entity may be permitted to accept tasks or access services only when its trust level satisfies the required security level. Unlike traditional rule-based or linear-weighted scoring, trust inference is often non-linear and time-varying, because trust is continuously updated through repeated interactions via reward/penalty (reputation) mechanisms. Accordingly, deep learning has been explored for credit evaluation due to its ability to learn compact representations from high-dimensional evidence and capture evolving behavioral patterns. Privacy constraints in IoT further motivate federated learning, enabling model training without directly centralizing sensitive user/device data. Moreover, recent trust-evaluation frameworks incorporate deep reinforcement learning to adaptively optimize trust-related parameters (e.g., aggregation weights or incentive factors) under dynamic environments [9]. Nevertheless, many deep models still depend on reliable labels or carefully curated supervision signals, and their robustness may degrade under class imbalance, domain shift, or previously unseen attacks, which are conditions that frequently arise in real-world IoT deployments. These limitations motivate label-free or weakly supervised designs that remain effective when ground-truth risk labels are unavailable.

Consequently, quick and accurate identification of unknown risks is essential to ensure secure device collaboration and data exchange. However, the diversity of attack strategies and usage behaviors in IoT ecosystems significantly increases the difficulty of risk identification. Conventional risk assessment systems, which rely on predefined rules and expert knowledge, often struggle to adapt to unknown attacks in rapidly evolving threat landscapes [10–13]. Furthermore, feature relevance can vary substantially across IoT devices and applications, complicating model design and hindering generalization. For example, differences in power-consumption modes and data-transmission frequency may alter feature distributions and affect evaluation accuracy. Finally, the rapid growth in the number and heterogeneity of IoT devices exacerbates the challenges of scalable data processing and analysis.

To address these challenges, machine learning techniques have been increasingly applied to enhance threat detection and mitigation [1,14–16]. Machine learning algorithms can analyze large datasets, predict anomaly behaviors, and identify attack patterns, thereby overcoming the limitations of predefined rules in risk identification. Supervised learning models can differentiate between normal and abnormal states through extensive analysis of labeled data, while unsupervised learning models detect anomaly patterns

based on the unique characteristics of the data, thus protecting against potential threats, including zero-day attacks.

In this paper, we design an effective trust assessment mechanism that operates without labeled data, which is crucial for ensuring secure collaboration and data exchange in IoT environments. The proposed framework leverages machine learning to classify and analyze device behaviors, enhancing trust assessment in situations where mutual trust between devices is lacking. By applying unsupervised learning techniques, this framework facilitates anomaly detection and strengthens IoT system security, even against unknown attacks.

2 Releated Work

2.1 Artificial Intelligence (AI) in Risk Assessment

Jayasinghe et al. (2018) [17] employed a machine learning-based trust computation model to enhance the trustworthiness of IoT services by calculating the interaction frequency, work relationships in the service domain, and the distance and reward systems between collaborating parties, measuring device trust attributes. Subsequently, Principal Component Analysis (PCA) was used to compress feature dimensions and extract key features from the obtained trust attributes. These trust features were labeled using the K-means algorithm and ultimately used to train an SVM-based trust prediction model that accurately identifies the trust boundaries of any interaction to obtain optimal parameters for the final trust value.

Yang et al. (2022) [18] proposed a Generative Adversarial Learning trust management method suitable for 6G wireless networks, addressing the current shortage of relevant anomaly data in 6G networks to define trust judgment rules. Thus, combining fuzzy logic to compute trust attributes like packet loss rate, delay rate, and tampering rate, the input trust attributes are converted into low, medium, and high two-category fuzzy sets of trustworthiness, while mitigating trust uncertainty. Subsequently, using K-means, all feature vectors were clustered into three trust groups corresponding to high, medium, and low trust levels, and through a Generative Adversarial Networks (GANs) based Autoencoder, the feasibility of intelligent trust management was learned and analyzed. Their method effectively trained the network model to identify and adapt to new threat patterns through simulated interactions between attackers and defenders. In underwater wireless sensor networks, traditional trust management mechanisms are often difficult to apply due to the unique and challenging environment.

Jiang et al. (2020) [19] proposed a dynamic trust assessment and update mechanism based on the C4.5 decision tree, using fuzzy theory to quantify communication quality, energy consumption, and packet information into three levels of trust values. This mechanism includes a sliding time window for trust updates with reward and penalty factors. Trust values increase for trustworthy behavior and decrease for poor or malicious behavior.

Han et al. (2019) [20] improved detection of unknown attack patterns, times, and locations in underwater environments using K-means clustering. They trained an SVM classification model based on clustering results to detect malicious nodes, highlighting the effectiveness of machine learning algorithms in identifying patterns from historical data.

Ali-Eldin (2022) [21] in social IoT, proposed a machine learning-based trust management system, designing a framework that evaluates the trustworthiness of nodes in the network through Direct Trust Metric (DTM) and Indirect Trust Measures (ITM). Using K-means algorithm, nodes were clustered based on the combined score of direct and indirect trust from social interactions, distinguishing between trustworthy and untrustworthy nodes.

Sagar et al. (2023) [22] simulated human social relationships to assess the characteristics of trust between devices, using specific social characteristics of objects in SIoT (in terms of direct trust metrics, reliability, benevolence, credible recommendations, and relationship levels). They envisioned a trustworthy object classification framework and addressed the issue of trust feature weighting using the Artificial Neural Network (ANN) algorithm. Finally, an ANN-based trust assessment model was used to make trust decisions for devices requesting collaboration, identifying their trustworthiness (trustworthy, neutral, or untrustworthy).

Si et al. (2022) [23] proposed an improved Zero Trust Access Control Model for remote work and telecommuting scenarios based on clustering algorithms. This model considers parameters such as IP addresses, ports, and service types of node data packets during the clustering process. The clustering results are used as business security assessment indicators, participating in the assessment of the security levels of nodes and services, to allocate appropriate security policies.

2.2 Applications of Zero-Knowledge Proof (ZKP) in AI

Zhang et al. (2020) [24] proposed the ZKDT Scheme using the structure of the Merkle tree to ensure that the predictive path of the Decision Tree model has not been tampered with. They incorporated each step of the Decision Tree prediction process into the Merkle tree, where each node's data includes its position in the decision process and the decision outcome. A complete verification path is formed by linking data nodes, using Merkle's properties to verify the Decision Tree model architecture, and using the Aurora protocol in zk-snarks to generate a proof of their Decision Tree.

Liu et al. (2021) [25] proposed a verification method based on CNNs, zkCNN Scheme, which is based on a modified sumcheck protocol using the Fast Fourier Transform to improve the proof cost of convolution computations in CNNs and reduce the number of interactions and verification process. This significantly lowers the computational cost and time consumption caused by extensive calculations in deep learning, while also using the protocol to protect the model parameters of the CNN architecture (convolutional layers, ReLU functions, and pooling), inputs and outputs, providing a comprehensive CNN verification solution.

Singh et al. (2022) [26] proposed a new scheme, Singh, to address common multiplication and memory access issues in machine learning models using a consistent memory access structure to improve memory efficiency during multiplication operations. In traditional machine learning models, multiplication and memory access are often performance bottlenecks. Singh reduces the number of multiplication gates in the model circuit design and improves memory issues, while practically applied in verifying Decision Tree structures, through node, decision path length, and predicted feature quantity.

Ghodsi et al. (2017) [27] introduced a verification framework called SafetyNets, which executes the verification of deep neural networks (DNN) in an untrusted cloud computing environment. Since DNN inference is a compute-intensive task often outsourced to cloud services, the framework aims to address fundamental trust issues using an Interactive Proof (IP) completed through the sum-check protocol, verifying its input and output based on the function (like softmax sigmoid) used in the last layer of DNN. However, as it must provide related weights, it only guarantees Completeness and Soundness without achieving Zero-knowledge.

Huang et al. (2022) [28] designed the zkMLaaS Scheme to address when model training is entrusted to third parties to handle large-scale data and complex computational tasks. Customers wish to verify the integrity of their model training. The Scheme uses random sampling to verify hyperparameters and training cycles during the training process, thereby proving the training process's integrity without revealing the service provider's knowledge assets and training plan details, ensuring that customers can verify the integrity

of model training while preventing the server from potentially using backdoor attacks to compromise training data or modify model weights during the training process.

2.3 Summary

In recent research, we have found that fuzzy theory and clustering algorithms can be effectively applied in risk analysis and differentiation. Fuzzy theory primarily deals with uncertainty and ambiguity by transforming risk factors into computable and analyzable data through fuzzy sets and rules, thereby providing flexible and adaptive risk assessments. However, this theory is highly subjective, since the definition of sets and the formulation of rules are based on expert experience, which may lead to biases. On the other hand, clustering algorithms identify potential risk groups and patterns by grouping large datasets based on similarity through data mining techniques, which aids in formulating targeted risk management strategies. AI models not only contain valuable expertise but also process a large volume of sensitive data. Currently, most companies offering AI model services do so through API integration, but this method does not guarantee the accuracy of the results or the security of the model. However, the combination of ZKP technology offers effective solutions for protecting intellectual property and data privacy.

- Private Model with Public Input: In this mode, the computational parameters or logic of the AI model are protected for privacy, and only a proof is provided externally to validate the effectiveness of the results. For example, in the financial services industry, credit assessments can ensure that the credit reports provided are accurate without disclosing the assessment algorithm.
- Public Model with Private Input: This mode allows users to keep their data private while using a public model for computation. The system provides proof to confirm the correctness of the computation process, thereby eliminating the need to reveal specific input data to third parties. This method is particularly suitable for processing highly confidential data, such as medical record analysis.

3 Proposed Framework

In this section, we explain and introduce detailed components of the framework, the design process, and implementation steps of the risk engine, the process of integrating the ZKP technology with the risk engine to build the ZKP model, and the process for conducting zero-knowledge trust assessments between devices.

3.1 Overall Architecture

We propose a framework designed to ensure that devices remain safe and secure after authentication in a ZTA environment as show in Fig. 1. This framework consists of an IoT server and multiple IoT devices. Within this architecture, we incorporate the core principles of ZTA, where devices as equals and do not have inherent trust relationships. Any device could potentially be a security threat. The IoT server acts as an impartial third party, primarily responsible for establishing a Risk Engine to evaluate recent device traffic risks and integrating ZKP technology to enhance device security and protect model privacy, which we refer to as the zk-Risk Engine.

The zk-Risk Engine is the core of our architecture, using ZKP technology to establish trust relationships between devices. Before any data exchange or collaboration begins, participating devices must undergo a risk assessment and verify each other. This ensures that even in the absence of inherent trust between devices, trust relationships can be established through ZKP technology. It guarantees that the device is secure. Besides, it also prevents the risk assessment results and model structure from being tampered with, ensuring the accuracy and integrity of the assessment results and model.

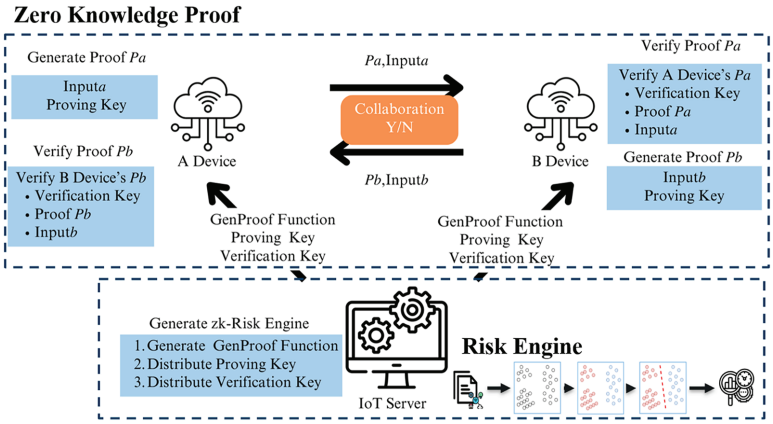

Figure 1: Overall architecture.

For example, in a Bring Your Own Device (BYOD) environment, employees use personal devices to work in the company office, at home, or in other public places, accessing the company's internal network and resources. After these devices complete authentication, they must undergo risk assessment by the zk-Risk Engine. Only after the device trust evaluation confirms that they are safe and normal can they connect to the company network. During the risk assessment process, both devices use ZKP technology to verify each other's security. Only devices that are assessed as secure can access sensitive company data, ensuring the safety of company resources.

3.2 Building of Risk Engine

Our risk engine is designed to build a continuous and dynamic trust assessment mechanism for interactions between IoT devices, with the main objective of identifying and assessing potential risks from the packet Information of recent interactions between devices. This mechanism is implemented through advanced machine learning techniques to ensure the system can maintain its security and reliability even in the face of unknown threats. As shown in Fig. 2, the implementation of the risk engine is divided into three main processes: Feature selection, Clustering and Labeling, and Calculating feature weight.

- Feature selection stage: The process of filtering out key features from a large amount of data that are useful for risk prediction. This process not only removes unnecessary data but also ensures that the model focuses on the key features that can most affect the risk assessment, which in turn improves the overall prediction accuracy and efficiency. In addition to excluding features, we use the MinMaxScaler, a common data preprocessing method that retains the original distribution of the data while scaling all features to a range between [0, 1]. This normalization ensures that the influence of each feature during model training is balanced, mapping them to a specific range.
- Clustering and Labeling stage: In this stage, feature vectors are clustered into two groups using the K-means algorithm, categorizing them into trustworthy and potentially anomalous. Although K-means implicitly favors spherical partitions and may not be optimal for all network-traffic patterns, we adopt it intentionally as an extremely lightweight (compared to DBSCAN), label-free baseline to provide an initial separation of interaction records. This baseline design enables us to isolate and evaluate the contribution of unsupervised clustering to the overall pipeline before introducing more complex clustering models. Each packet message is grouped based on its feature distance from the average of each cluster. This precise tagging of trust levels for each packet during interactions facilitates accurate labeling. The labeled information is then used to construct a training dataset, with the trust level of

these packets classifying the device state as "trusted" or "potentially anomalous," based on the centroid locations of the clusters. Moreover, we applied the Elbow Method to determine the optimal number of clusters, thereby improving the clustering performance. The Elbow Method involves calculating the Sum of Squared Errors (SSE) for different numbers of clusters and plotting the SSE against the number of clusters. As the number of clusters increases, the decrease in SSE gradually slows down, forming a shape similar to an elbow. We select the number of clusters corresponding to the elbow point as the optimal value.

- Calculate feature weights stage: After labeling the dataset as trustworthy and potentially anomalous, we use the Decision Tree algorithm to train the risk assessment model. We use the Decision Tree algorithm to train the risk assessment model. Our Decision Tree model is constructed using Gini Impurity to assess and incrementally split the dataset. The formula is as follows: For a given training set D, which contains K different categories, where p_k is the proportion of category k in the node, the Gini Impurity G of the set is defined as:

$$G(D) = 1 - \sum_{k=1}^{K} p_k^2$$

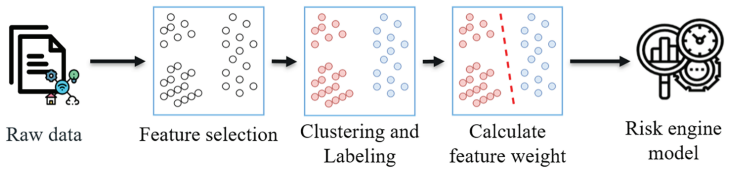

Figure 2: Steps to build the risk engine.

When evaluating a specific split, the algorithm calculates the weighted Gini Impurity for each child node and seeks the split that minimally reduces impurity. Specifically, the algorithm chooses splits that minimize the following weighted sum where $|D_1|$ and $|D_2|$ are the sample sizes of the left and right child nodes after the split, $G(D_1)$ and $G(D_2)$ are the respective Gini Impurities of the child nodes, and $|D|$ is the total sample size of the original node:

$$G_{\text{split}} = \frac{|D_1|}{|D|} G(D_1) + \frac{|D_2|}{|D|} G(D_2)$$

The Decision Tree was chosen due to its simple structure and transparent decision-making process. Additionally, its training and operation do not require many computational resources, making it very suitable for environments with limited computing capabilities. Through the Decision Tree analysis, we determine which features have a decisive impact on risk assessment and their corresponding weights. Subsequently, these features and weights are combined with ZKP technology to enhance risk assessment privacy and security.

3.3 Generation of zk-Risk Engine

We implement the inference component of ZKDT [24], which commits to a trained Decision Tree (DT) using a Merkle Hash Tree (MHT) so that a verifier can check that the reported prediction was produced by an *untampered* model. An MHT is a binary hash tree where each leaf stores the hash of a data record and each internal node stores the hash of the concatenation of its children; the root hash is therefore a succinct

commitment to all underlying node data. During verification, recomputing the root from authenticated paths and comparing it with the public root ensures integrity of the committed model.

Step 1: Circuit-friendly representation of continuous thresholds. A DT split is a real-valued predicate of the form $x \leq \theta$. To obtain a canonical encoding suitable for both hashing and ZK circuits, we apply fixed-point quantization. For each learned threshold $\theta \in \mathbb{R}$, we store an integer $\Theta = \lfloor 100 \cdot \theta \rceil$, and for each feature value x at inference time we compute $X = \lfloor 100 \cdot x \rceil$. Each predicate $x \leq \theta$ is then evaluated as the integer comparison $X \leq \Theta$. This avoids ambiguity caused by floating-point representations and ensures the same parameters are used in (i) the ZK comparison constraints and (ii) the hash payload committed in the MHT.

Step 2: Missing values. After preprocessing, our datasets contain no missing values. We apply a deterministic data-cleaning pipeline before performing the train/test split; therefore, the DT inference circuit does not require any missing-value routing policy (e.g., default-branch handling).

Step 3: Serialize the DT into an Authenticated Decision Tree (ADT). To commit the model, we first serialize the DT into an ADT structure (Fig. 3). Each node v_i is assigned a unique index i and stored with its structural position and parameters. For an internal node, we store:

$$v_i = \langle i,\ \ell_i,\ f_i,\ \Theta_i,\ i_L,\ i_R \rangle, \tag{1}$$

where ℓ_i is the depth (level), f_i is the feature identifier, Θ_i is the quantized threshold, and i_L, i_R are the indices of the left and right children. For a leaf node, we store:

$$v_i = \langle i,\ \ell_i,\ y_i \rangle, \tag{2}$$

where y_i is the predicted class/risk label.

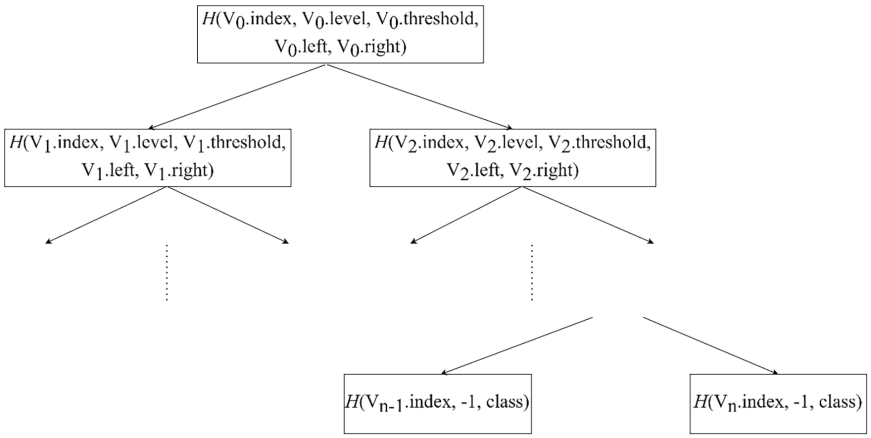

Figure 3: Authenticated decision tree structure.

Step 4: Handle irregular branches (padding to a full binary tree). Learned DTs are typically unbalanced, i.e., leaves can occur at different depths. Variable-depth paths would otherwise lead to input-dependent execution and non-uniform proof costs. To ensure consistent hashing and predictable proving time, we compile the DT into a fixed-shape full binary tree up to a maximum depth D by padding shorter branches with dummy nodes. In our implementation, we set $D = 7$, resulting in $2^{(D+1)} - 1 = 255$ total node positions. A dummy node is marked by $\ell_i = -2$ and deterministically forwards to the same leaf label, i.e.,

$$v_i = \langle i,\ -2,\ y_i \rangle. \tag{3}$$

Padding preserves the original DT output while guaranteeing a constant number of comparisons per inference.

Step 5: Construct the Merkle Hash Tree commitment. We compute a leaf hash for each ADT node using a canonical encoding:

$$h_i = H(\text{encode}(v_i)). \tag{4}$$

Internal Merkle hashes are computed bottom-up as:

$$h_p = H(h_L \,\|\, h_R), \tag{5}$$

until the Merkle root h_{root} is obtained. The model commitment is then computed as:

$$C = H(h_{\text{root}} \,\|\, r), \tag{6}$$

where r is a random nonce. The commitment C (or equivalently (h_{root}, r)) is published, and any modification of DT structure or thresholds changes h_{root} and is therefore detectable. For example, in Fig. 4, each leaf node H_1, H_2, H_3, and H_4 represents the hash values of A, B, C, and D, respectively. These values are further merged into the next layer of nodes, such as H_1 and H_2 merging to form node $H_{1,2}$, and similarly, H_3 and H_4 merge to form node $H_{3,4}$. This merging process continues until the root node $H_{1,4}$ is formed.

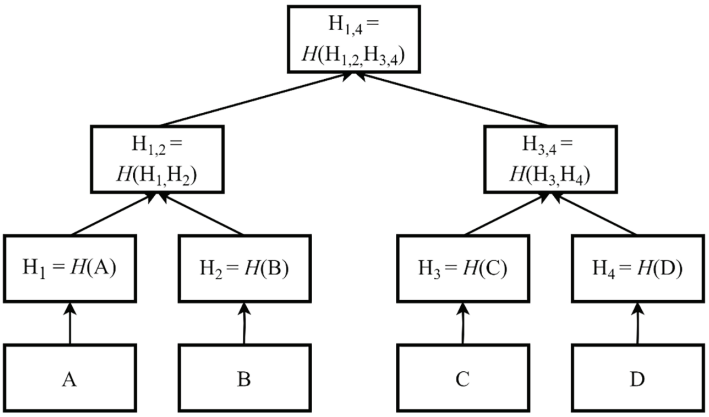

Figure 4: Merkle hash tree structure.

Step 6: What the proof verifies. Given an input feature vector, the prover evaluates the DT (using the quantized comparisons) and provides (i) the predicted label, and (ii) the Merkle authentication paths for the nodes on the evaluated decision path. The ZK proof enforces that the branch decisions are consistent with the quantized thresholds and that all visited nodes are valid members of the committed MHT root. As a result, the verifier is convinced that the output was produced by the committed DT without revealing the full tree or intermediate decision details beyond what is required by the protocol.

Next, we construct an MHT over all ADT node records and obtain the Merkle root h_{root}. We then publish the model commitment as $\text{com}_r = H(h_{\text{root}} \| r)$, where r is a random nonce. Any modification to the DT structure or its quantized thresholds changes h_{root} and therefore invalidates com_r.

Finally, we implement the ZKDT circuit in Circom [29]. The public inputs include the quantized input features d, the predicted output y_a, and the commitment com_r. The private witness includes the DT decision-path indices path_a, the Merkle authentication siblings along that path, and the nonce r. The prover generates

π_{ZKDT} by proving (i) the split comparisons along path_a are consistent with the quantized thresholds, and (ii) the corresponding ADT node hashes are valid members under the committed root (via the provided siblings). The verifier recomputes the root from the authenticated path inside the proof and accepts if it matches com_r. The initial parameter settings as follows:

- $PP \leftarrow \mathbf{ZKDT.G}(1^{\lambda})$ Choose a collision-resistant hash function.
- $\mathbf{com}_r \leftarrow \mathbf{ZKDT.Commit}(T, PP, r)$ Hash leaf nodes pairwise up to the tree root, and calculate the hash value with a random value r.
- $\pi_{\mathbf{ZKDT}} \leftarrow \mathbf{ZKDT.P}(T, \mathbf{path}, PP)$ Given the Decision Tree T and a path, π_{ZKDT} is computed using the path along with sibling nodes and the random value r.
- $\{0,1\} \leftarrow \mathbf{ZKDT.V}(\mathbf{com}_r, \mathbf{path}, \pi_{\mathbf{ZKDT}}, PP)$ Given path and π_{ZKDT}, recompute the hash values along path and π_{ZKDT}, and compare the root hash value with π_{ZKDT}. If they are the same, output 1; otherwise, output 0.

Additionally, as shown in Fig. 5, the public information of the circuit parameters includes input data d, output y_a, com_r, and the private information path indices d', path_a, siblings, random number r, and the $expdate$ used to generate and verify the proof's expiration date.

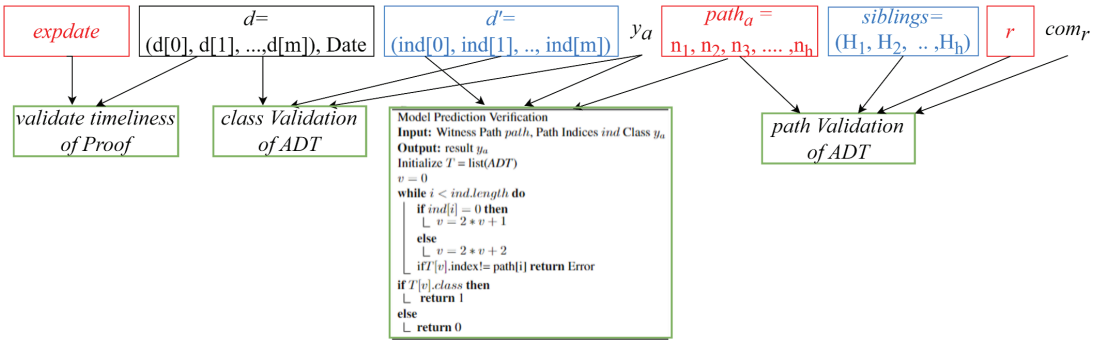

Figure 5: Circuit parameters and generation of private information for proof of ZKDT.

In Algorithm 1, the Generate Witness Algorithm generates a witness through a process that utilizes specific attributes of the data sample to traverse the ADT layer by layer, identifying the path taken at each level. Simultaneously, it records the sibling nodes and path indices associated with the current node. Subsequently, the algorithm performs hash operations on each node to construct the ADT based on a Merkle Tree structure. This process includes hashing based on the information of both leaf and non-leaf nodes, as well as a random number r, thereby generating the model's commitment com_r. Finally, the algorithm retrieves the corresponding path and sibling nodes within the Merkle Tree structure's ADT based on the index of the predicted path and the index of the sibling nodes.

Finally, we use the Groth16 protocol [30] to generate the relevant proofs. Groth16 is part of the Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARK) systems, characterized by their extremely succinct proofs that are small in size and easy to verify, requiring no repeated challenges and responses between parties. Among all zk-SNARK protocols, Groth16 has the smallest proof size and is based on Quadratic Arithmetic Programs (QAP). Thus, it transforms the computational approach of our risk assessment model into polynomial problems and generate ZKP involving QAP relations. After completing the ZKDT circuit description, preliminary Groth16 setup tasks must be performed, generating a trusted setup for each circuit. Upon completing this step, we obtain the model's corresponding proving key (pk) and verification key (vk). Therefore, IoT devices can use ZKDT to generate a witness and proof using pk, creating

a post-evaluation proof π. Any device can then use the public vk, along with the public information, to verify the proof results π. The Zhang et al. ZKDT was adopted as an engineering-light, conservative baseline for our first end-to-end zk-Risk Engine prototype. ZKDT requires only fixed-point split comparisons with hash constraints to authenticate the visited decision-path nodes against the public Merkle commitment, yielding predictable proving/verification costs that scale primarily with the tree depth (which we fix via padding). Newer ZKDT frameworks based on matrix-lookup commitments can significantly reduce proving time, but they add non-trivial protocol and preprocessing complexity. We therefore use ZKDT for this initial study and leave a direct benchmark against matrix-lookup ZKDT as future work.

3.4 Framework Initialization and Collaboration Process

In this section, we describe the initial setup of our proposed framework and the steps for conducting a trust evaluation before collaboration between devices.

- **Initial Phase:** In the initial stage, all devices are assumed to have passed authentication, making their historical traffic and behavior data trustworthy. The IoT server will collect this data and use it to establish the zk-Risk Engine, which defines the device risk evaluation model. During this process, the server converts the risk model into an arithmetic circuit, referred to as the GenProof Function, then executes the zk-SNARK setup algorithm to generate a pk (used by devices to generate risk proofs) and vk (used by other devices or server to verify proofs). Once generated, the server securely distributes those GenProof Function, pk and vk to all registered devices (demonstrated in Fig. 6). To ensure the timeless and prevent replay or long-term misuse, each of these components includes a validity period, defined by timestamps (e.g., valid from DateTime0 to DateTime1). Devices can only generate valid proofs within this time window. After expiration, the server performs a new trusted setup to regenerate updated pk and vk, and redistributes them. Moreover, the pk, vk, and GenProof Function on the devices must be strictly protected to prevent unauthorized access. During the same time, each device will record traffic data and generate proofs during the collaboration between devices.
- **Collaboration Phase:** Before any data exchange or cooperation begins, both parties must build a secure channel using encryption protocols such as TLS to verify the identities of each other. Next, they must conduct risk assessments and validate each other to prove their secure state. As shown in Fig. 7, when Device A and Device B intend to collaborate, they will use the GenProof Function and pk to perform their respective risk assessments and generate proofs. These proofs, a public statement, and the current timestamp will be sent to the other party. The receiving device (or server) verifies the proof using the vk. Once both parties successfully verify the proofs provided by each other, they will be confirmed to be in a secure state, allowing a safe cooperation. Otherwise, the interaction is rejected. Additionally, devices can periodically check their zk-Risk Engine or keys through a self-check mechanism to ensure they have not been tampered with, and immediately issue an alert if any anomalies are detected, further ensuring the system's security.

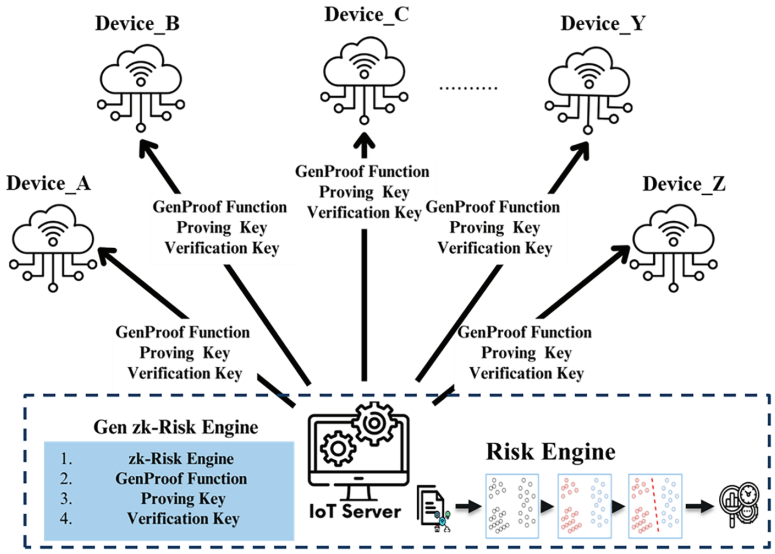

Figure 6: Initial setup.

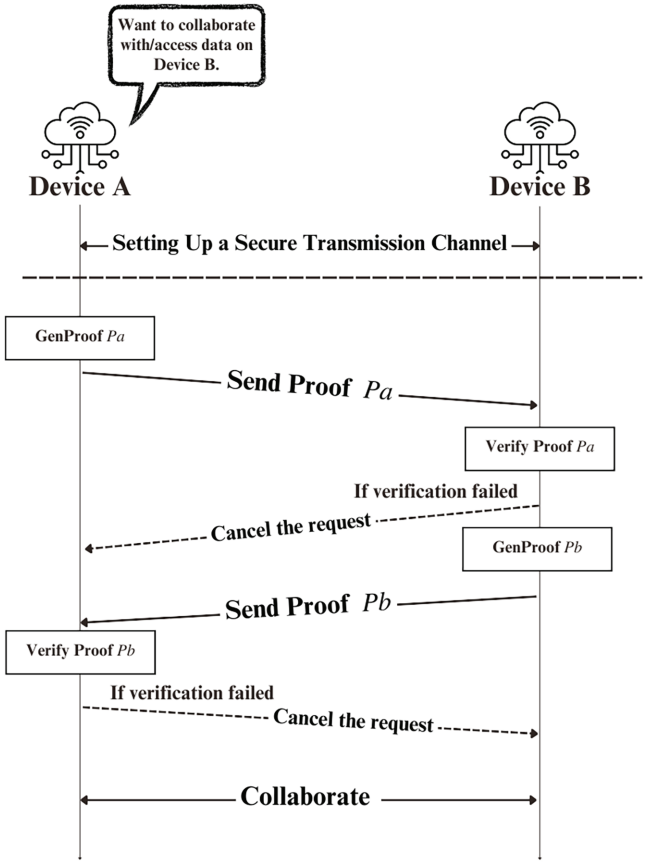

Figure 7: Collaboration flowchart.

Algorithm 1: Generate witness

Require: Data sample a

Ensure: Commitment (com_r), Class (y_a), Witness Path *(path)*, Siblings *(siblings)*, Random Number *(r)*, Path-Indices *(ind)*

$T \leftarrow list(ADT)$
$N \leftarrow T$.length, $i \leftarrow T$.length $- 1$
$HashADT \leftarrow$LIST(N)
$r \leftarrow$RANDOM()
$j \leftarrow a$.length $- 1$
$expiryDate \leftarrow$ TIMESTAMP
if $(a$.Date$) > expiryDate$ **then**
    return Error
end if
$cur \leftarrow 0$
while $j \geq 0$ **do**
    $path$.APPEND(cur)
    if $a[j] \leq T[cur].thr$ **then**
        $siblings$.APPEND(cur . 2 + 2)
        $cur \leftarrow 2 \cdot cur + 1$
        ind.APPEND(0)
    else
        $siblings$.APPEND(cur . 2 + 1)
        $cur \leftarrow 2 \cdot cur + 2$
        ind.APPEND(1)
    end if
    $j \leftarrow j - 1$
end while
REVERSE($path, siblings$)
$y_a \leftarrow T[cur].class$
while $i \geq 0$ **do**
    if $T[i].level = -1$ **or** $T[i].level = -2$ **then**
        $HashADT[i] \leftarrow H\big(T[i].index \parallel T[i].class\big)$
    else
        $HashADT[i] \leftarrow H\big(HashADT[2i+1] \parallel HashADT[2i+2] \parallel T[i].index \parallel T[i].level \parallel T[i].thr\big)$
    end if
    $i \leftarrow i - 1$
end while
$com_r \leftarrow H\big(r \parallel HashADT[0]\big)$
for $i = 0$ **to** $path$.length **do**
    $path[i] \leftarrow HashADT[path[i]]$
    $siblings[i] \leftarrow HashADT[siblings[i]]$
end for
return com_r, y_a, $path$, $siblings$, r, ind

3.5 Connection between Proposed Method and ZTA Tenets

To explicitly trace the proposed zk-Risk Engine framework to the ZTA tenets in NIST SP 800-207, we map each tenet to concrete components and workflow steps within the architecture. For Tenet 1, every participating IoT device and service endpoint is treated as a resource whose behavior must be evaluated rather than implicitly trusted; accordingly, the IoT server and all devices are onboarded into the same risk-assessment and proof workflow and begin with no default trust relationships. For Tenet 2, the same trust procedure is enforced regardless of where a device is connected from because the collaboration phase first establishes a secure channel and then performs proof-based trust validation before any data exchange, so the decision logic never depends on being "inside" a perimeter. For Tenet 3, trust is tied to each interaction: before starting a session, both parties execute risk assessment, generate proofs locally, exchange them, and verify each other to proceed, which anchors trust in the current request context rather than in a one-time enrollment. This per-session trust signal is intentionally separated from authorization scope: in our deployment model, fine-grained permissions and least-privilege enforcement remain the responsibility of the server-side administration layer (e.g., OS services, access-control rules, or a policy engine/PEP), which can consume the zk-Risk Engine's accept/deny output and risk evidence as an input when deciding what actions a verified session is allowed to perform. For Tenet 4, the "dynamic policy" aspect is realized by the risk engine's runtime decision based on observable device and traffic attributes; concretely, we derive the attribute vector from traffic metadata, train the unsupervised labeling and Decision Tree classifier, and then record the resulting feature thresholds that are embedded into the GenProof/Verify workflow, yielding an auditable, model-grounded policy condition rather than ad-hoc rules. This also addresses the reviewer's concern on feature selection: the Decision Tree gives an explicit importance ranking and threshold structure, so the final feature subset is justified by learned split criteria and can be reported as part of the policy definition, rather than chosen arbitrarily. For Tenet 5, continual monitoring is implemented at the transaction level: every subsequent communication is subjected to the same proof generation and verification gate, and the sliding window makes this monitoring real-time by constraining decisions to the most recent W traffic points, ensuring that older behavior does not dominate the current posture and allowing rapid response to abrupt changes or attacks. For Tenet 6, strict enforcement and reauthentication triggers are supported because proof verification is an explicit gate in the collaboration flow and the design includes a self-check mechanism to detect tampering of keys or the zk-Risk Engine; when a session's behavior is classified as anomalous or malicious, the server can deny the request and require re-authentication or re-attestation before allowing further requests. For Tenet 7, we close the loop by using the information already collected by the framework to improve posture over time: the server aggregates traffic metadata and prior risk outcomes from ongoing sessions, then periodically recalibrates the window size W, retrains or updates clustering/Decision-Tree thresholds under drift, and republish updated proving/verification parameters with explicit validity periods via the existing key distribution mechanism, so the trust decision evolves using the latest observed device and network state and directly informs future policy enforcement.

4 Experiments

In this section, we utilize two different datasets to conduct experimental evaluations of our proposed Risk Engine framework while integrating ZKP technology. Afterwards, we assess the performance of the Risk Engine and verify the effectiveness of ZKP.

4.1 Performance of Risk Engine

In our model selection process, we evaluated three common machine learning algorithms: Logistic Regression (LR), Support Vector Machine (SVM), and Decision Tree. To comprehensively assess the

performance of each algorithm in handling trust decision tasks, we chose to use elements of the confusion matrix as the main observational metrics. These metrics include Precision, Recall, and F1 Score. These three indicators provide information about the model's performance in correctly predicting positive and negative cases, allowing us to thoroughly analyze and compare the strengths and limitations of each model. The formulas for these metrics are as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{7}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{8}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{9}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}$$

- **Accuracy:** Indicates the ratio of correct predictions to total predictions. This is the most intuitive and comprehensive metric, but may not reflect the overall model's performance, especially with imbalanced datasets.
- **Precision:** Represents the proportion of actual positives among all cases predicted as positive by the model. High precision means that the model produces fewer false positives when predicting positive cases.
- **Recall:** Measures the proportion of positive cases identified by the model out of all actual positive cases. This reflects the model's ability to capture true positive events. In trust models, high recall is crucial, as it indicates that the model can identify and correctly handle most situations that should be trusted, which is vital to avoid missing any potential anomalies.
- **F1 Score:** F1 Score is the harmonic mean of Precision and Recall, serving as a balanced metric between the two. In situations where Precision and Recall are equally valued, the F1 Score provides a single measure to evaluate the overall effectiveness of the model.

4.2 Case 1—CIC-IoT2023 Datasets

The CIC-IoT2023 datasets [31], created by a Canadian cybersecurity research institute, feature a network topology of 105 IoT devices within a smart home environment and simulate 33 different types of attacks, including DDoS, DoS, Recon, Web-based, Brute Force, Spoofing, and Mirai. These datasets meticulously record the packet traffic data of both normal and malicious IoT devices attacking other devices. Among the 47 features provided in CIC-IoT2023, we used a subset of 20 quantitative flow-level features and excluded protocol indicators and TCP status-flag features to reduce dimensionality and avoid sparsity/noise in downstream analysis. Specifically, the retained features are: *flow_duration, Duration, Rate, Srate, Drate, ack_count, syn_count, fin_count, urg_count, rst_count, AVG, Std, Tot size, IAT, Number, Magnitue, Radius, Variance, Weight, and Covariance*, which are typically sparse and stack-dependent. We select this final feature set via an empirical separability screening: K-means is executed repeatedly with different initializations/iteration budgets, and each feature is ranked by a centroid-based separation score (between-cluster variance normalized by total variance). Features with near-zero scores across runs are removed, and highly correlated variables are pruned to reduce redundancy. Finally, we report feature contributions to the risk engine using the decision tree by summarizing split frequency and impurity reduction of the retained features.

Before training trust models, it is necessary to label these packet flows for risk. We must extract features from the collected raw packet information to effectively process and analyze the data. Therefore, we exclude qualitative data, such as protocol types, from the original features, as such data is generally difficult to use

directly in quantitative analysis. We do not use these qualitative types as the main features in our primary analysis process. Instead, they serve more as auxiliary tools, while our main focus is on indicators that reflect quantitative changes. The key characteristic information includes data such as traffic size, packet intervals, and rates, which helps us to statistically analyze recent traffic and rate data.

In our method, we performed unsupervised K-means clustering on the CIC-IoT2023 dataset, which contains 4,829,173 flow records and is provided with ground-truth labels that distinguish Benign traffic from multiple attack classes (e.g., DoS/DDoS, Recon, Web-based, Brute Force, Spoofing, Mirai). Although these labels enable supervised learning, we intentionally did not use them during clustering, and instead used them only as a reference for interpreting the discovered groups.

We selected the number of clusters K using the Elbow Method (Fig. 8). Because "risk level" is application-dependent and the dataset's labels indicate attack type rather than an explicit risk tier, we defined normal vs. anomalous states post hoc based on cluster centroids and their dominant flow characteristics. Specifically, clusters exhibiting longer flow duration and larger dispersion-related statistics tended to align with stable, benign-like behavior, whereas anomalous clusters showed more uniform packet-length variation within flows. In addition, malicious IoT behavior commonly generates a higher volume of incoming traffic, reflected by an elevated SRate. Based on these centroid-level patterns (Table 1), we labeled clusters 1, 2, and 3 as normal traffic, while clusters 0 and 4 were categorized as anomalies.

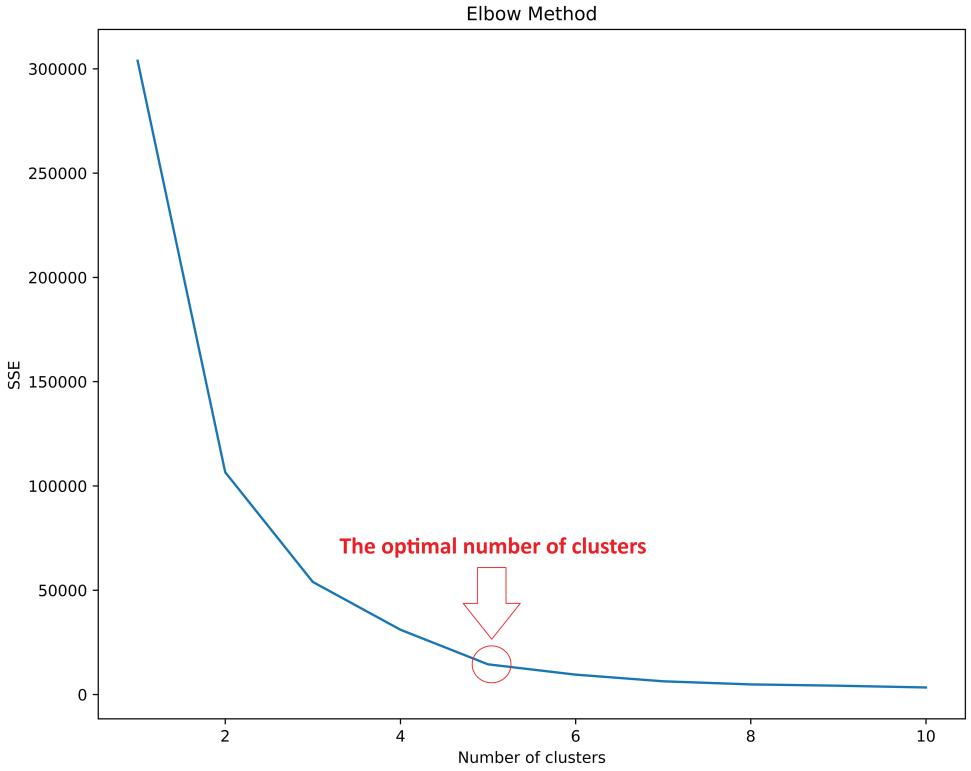

Figure 8: Case 1 optimal value for *K*.

In our analysis, Table 2 shows that most traffic data can be successfully classified into the correct group, with an accuracy rate of 98.96% for normal clustering and 95.39% for anomalies, resulting in an average accuracy of 95.47%. However, a few atypical traffic attack data points are misclassified as normal. For instance, some atypical traffic attacks (such as reconnaissance, spoofing, web, and brute force attacks)

are easily misclassified into the normal behavior group. Table 3 emphasizes strong performance on the dominant *Anomaly* class (Precision = 0.9998, Recall = 0.9541, F1-score = 0.9764), indicating that most predicted anomalies are correct and the majority of anomalies are detected. In contrast, the *Benign* class achieves very high Recall (0.9923) but low Precision (0.3427), suggesting that many samples predicted as benign are actually anomalous (a high false-negative rate for anomaly).

Table 1: Descriptive statistics of selected flow features.

Cluster	Flow_ duration	Duration	Rate	Srate	Drate	Ack_count	Syn_count	Fin_count	Urg_count	Rst_count	AVG	Std.	Tot size
0	0.505	64.122	9174.770	9174.770	0.0	0.103	0.317	0.110	0.699	2.383	73.908	0.163	73.973
1	122.296	108.306	2019.524	2019.524	0.0	0.050	0.683	0.047	134.561	880.737	592.036	554.555	590.398
2	124.103	109.265	1930.799	1930.799	0.0	0.049	0.683	0.051	137.356	892.943	591.823	331.976	592.949
3	11.561	69.220	9186.678	9186.678	0.0	0.029	0.268	0.060	8.847	33.024	770.664	475.287	770.878
4	0.776	65.893	10647.186	10647.186	0.0	0.055	0.315	0.070	0.187	0.741	134.242	11.238	134.024

Table 2: Post-hoc cluster–label agreement.

Attack Type	Orginal Count	Correct Clustering	Accuracy
Benign	118,697	117,461	98.96%
Anomaly	4,933,501	4,706,035	95.39%
Benign + Anomaly	5,052,198	4,823,496	95.47%

Table 3: Per-class evaluation metrics (Benign vs. Anomaly).

Class	Precision	Recall	F1-score
Anomaly	0.9998	0.9541	0.9764
Benign	0.3427	0.9923	0.5095

Next, Fig. 9 clearly illustrates that each split in a Decision Tree is based on a threshold value of a selected feature, and the choice of these threshold values directly impacts the depth and complexity of the decision tree. In our experiment, the depth of the decision tree is set to 7, while all other parameters are left at their default values in the scikit-learn framework. The layer-by-layer deepening of splits not only enables us to precisely identify which features are decisive in the final decision but also optimizes the model's computational efficiency. Each leaf node represents a final decision, indicating whether data points are trustworthy. This approach not only enhances the model's interpretability, allowing us to clearly understand the main factors influencing classification decisions, but also facilitates more efficient data storage and processing on IoT devices. By storing only these key features and decision thresholds within the high-dimensional feature space, we significantly reduce storage requirements. Finally, this model structure can be converted into an ADT structure and records each split's decision threshold for subsequent use with ZKP technology.

In the CIC-IoT2023 datasets, according to our multiple test statistics, as shown in Figs. 10–12, among 100,000 instances of normal behavior with varying proportions of malicious attacks added, the Decision Tree demonstrated superior classification accuracy compared to LR and SVM. This is because the Decision Tree model has advantages in handling nonlinear relationships and complex decision boundaries. Compared to LR and SVM, Decision Trees are better equipped to deal with data heterogeneity and outliers. Therefore,

when dealing with the CIC-IoT2023 dataset, which contains various types of malicious attacks, it provides higher classification accuracy.

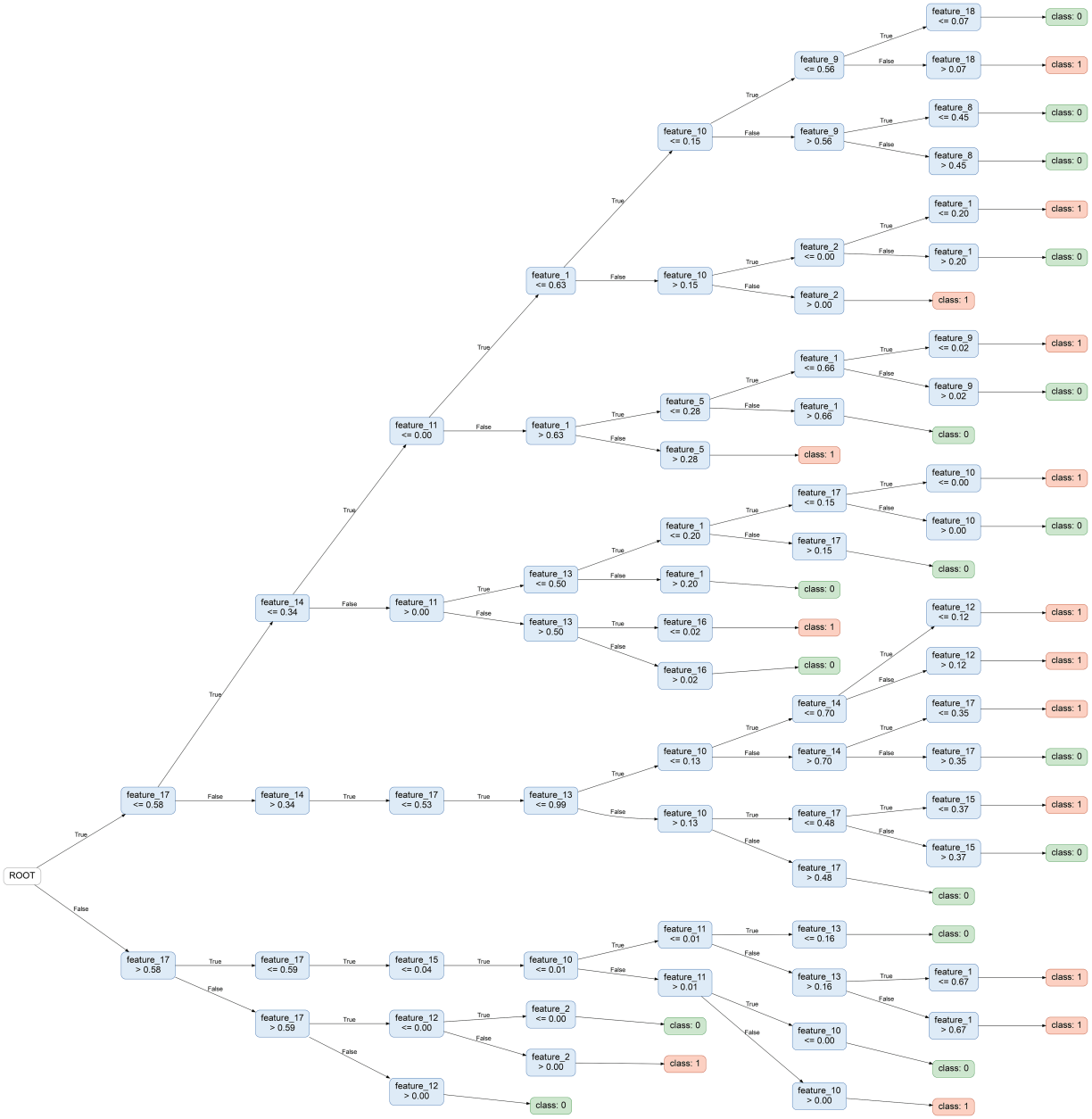

Figure 9: Decision tree split process description.

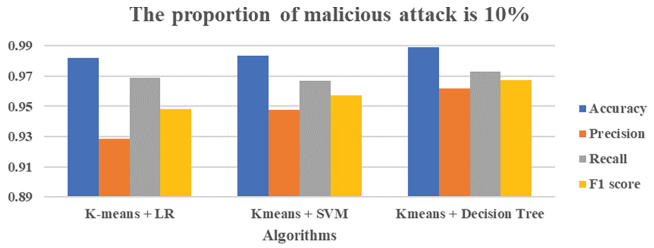

Figure 10: The proportion of malicious attack is 10%.

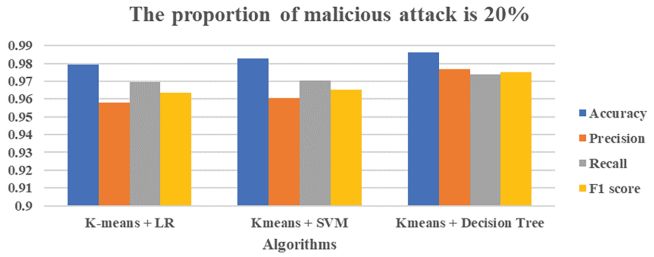

Figure 11: The proportion of malicious attack is 20%.

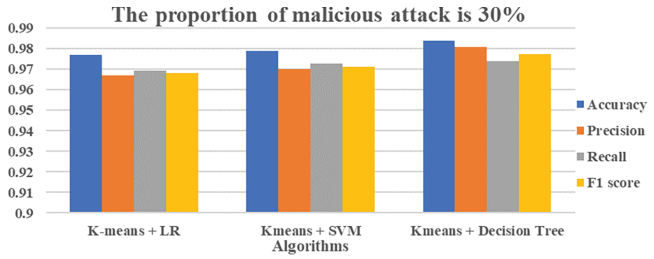

Figure 12: The proportion of malicious attack is 30%.

In Table 4, we recorded 200 experiments with different proportions of malicious attacks among 100,000 instances of normal behavior, using results from different time frames on the Decision Tree model. We compared the average performance over various numbers of recent traffic records from devices using a sliding window approach. Data analysis revealed that the number of recent records significantly impacts the model's performance. The model performed best when using the most recent five records, indicating that a higher volume of data records within a shorter time window enhances the model's prediction accuracy and reliability. However, as the number of records increased to seven or more, we observed a decline in both accuracy and recall rates. This suggests that an excessively high number of records may prevent the model from effectively capturing true threats.

Table 4: Comparison of different recency of data records.

Records	10% Attack		20% Attack		30% Attack	
	Accuracy	Recall	Accuracy	Recall	Accuracy	Recall
Last 3 records	91.37%	95.51%	84.50%	91.95%	78.30%	88.78%
Last 5 records	94.49%	92.88%	89.76%	91.27%	86.14%	89.30%
Last 7 records	96.55%	82.05%	92.43%	82.66%	88.82%	82.84%
Last 8 records	96.62%	70.08%	90.40%	72.76%	84.51%	75.00%
Last 9 records	96.01%	65.02%	87.49%	68.68%	79.69%	71.77%
Last 10 records	94.87%	59.51%	82.74%	63.11%	71.85%	66.48%

4.3 Case 2—Dataset from a Cybersecurity Company

Based on the Case1 experiment results, we conducted a real-world deployment using the device behavior dataset provided by a cybersecurity company in Taiwan. The reason for choosing this dataset is that no security incidents have occurred so far, meaning there is no ground truth available. This allows us to verify that our method can effectively differentiate risks and assess device trustworthiness. The contents of the dataset are shown in Table 5. This dataset contains 855,198 records and 11 columns, comprising device-authentication logs and contextual signals (e.g., switch/firewall behaviors and geographic coordinates). After applying PCA-based feature contribution analysis, we reduced the input dimensionality from 11 to 8 feature columns (Table 6) by removing attributes that contribute minimally to the dominant principal components (class, customerId, and ip).

We conducted K-means clustering on this dataset, which consists of 885 users. Using the Elbow Method, we identified that the optimal value for K (number of clusters) is 5 (Fig. 13), and we divided the dataset into 5 clusters, as shown in Table 7.

Table 5: Features of the dataset from a cybersecurity company in Taiwan.

#	Feature	Description
1	Time	Timestamp
2	Class	Types of APIs
3	icpId	ICP identifier
4	customerId	Customer identifier
5	userId	ICP unique generated identifier for the user
6	deviceId	Device model and number
7	ip	Devide IP
8	resultCode	Device authetification result
9	lat	The device's latitude
10	lon	The device's longitude
11	firewall	Firewall switch behavior

Table 6: Extracted features of the dataset from a cybersecurity company.

#	Feature	Description
1	Time	Timestamp
2	icpId	ICP customer identifier
3	userId	ICP unique generated identifier for the user
4	deviceId	Device model and number
5	resultCode	Device authentication result
6	lat	The device's latitude
7	lon	The device's latitude
8	firewall	Firewall usage habits ratio

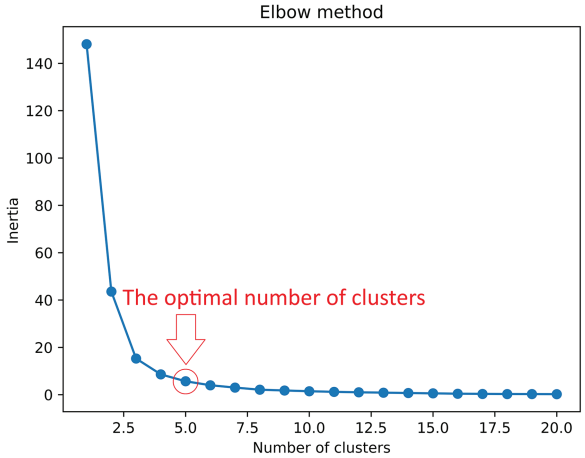

Figure 13: Case 2 optimal value for *K*.

Table 7: Case 2 cluster centers.

Cluster	Count	Failure	Other_err	Result Code_100	Result Code_101	Result Code_102	Result Code_103	Result Code_104	Result Code_105	Result Code_106
0	83	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	6	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2	42	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3	7	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
4	7	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0

Since ground-truth labels are unavailable for Case 2 due to the rarity of confirmed events, we adopt a clustering-driven pseudo-labeling strategy consistent with Case 1, where K-means grouping achieved >95% agreement with the available labels. Specifically, we transfer labels by interpreting cluster centroids using domain indicators recommended by the dataset provider, including *resultCode_100*, firewall values, and error flags (*failure*, *other_err*). We also defined normal and anomaly states based on the centroids, as shown in Table 7. In normal states, *resultCode_100*, which indicates authentication success, and firewall values tend to be larger, whereas in anomaly states, authentication failures, such as *failure* and *other_err*, and frequent firewall changes are observed. We can define the results of the first and fourth clusters as normal behavior, while the remaining clusters were classified as anomalies.

Finally, the Decision Tree algorithm is used to construct a risk engine. As explained in Case 1, Fig. 14 clearly illustrates that each split is based on a threshold value of a selected feature, and the choice of these threshold values directly impacts the depth and complexity of the decision tree. The layer-by-layer deepening process of splits not only allows us to identify which features are critical in the final decision-making precisely, but also optimizes the computational efficiency of the model. Each leaf node represents a final decision, clearly distinguishing whether the data points are trustworthy or not. This model structure can be transformed into a tree-like structure, and each split's decision threshold can be recorded for subsequent use with ZKDT technology.

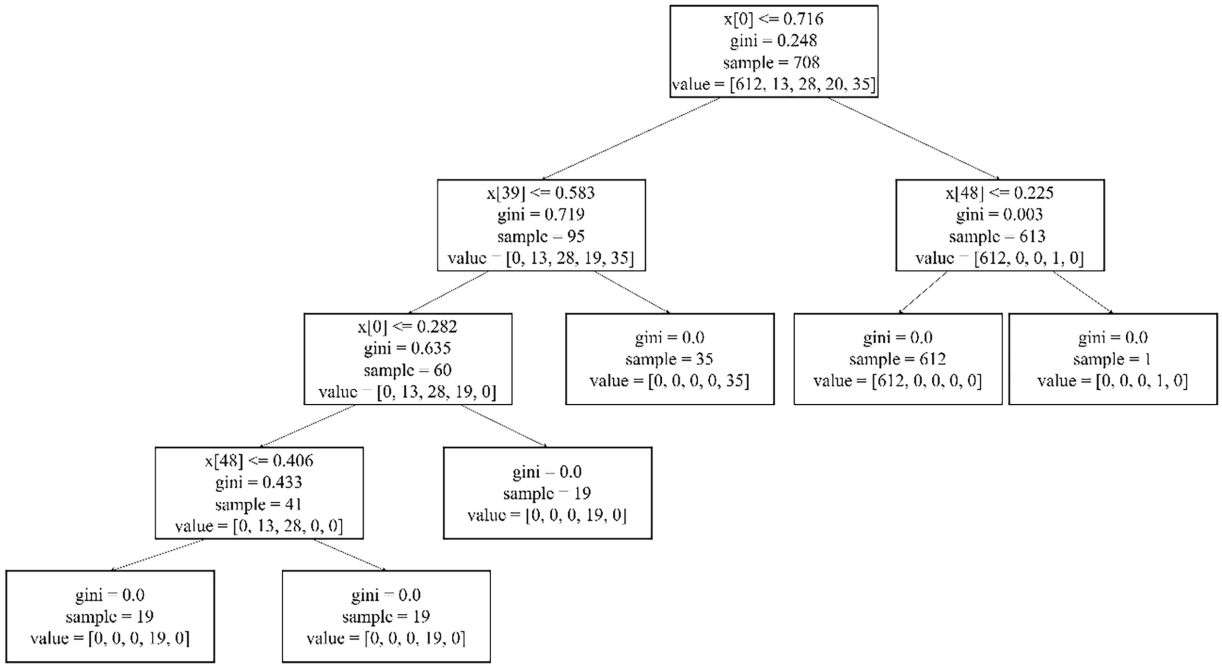

Figure 14: Case 2 decision features and thresholds.

4.4 Verification of ZKDT

On the ZKDT, we can ensure that the verification result y_a is the calculated outcome for the data sample d, assess whether the Decision Tree has been altered, and check the validity of the model prediction path. Based on the parameters for generating proofs, as shown in Figs. 15 and 16, we will discuss and validate through several cases:

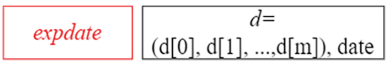

Figure 15: Circuit parameters of ZKDT.

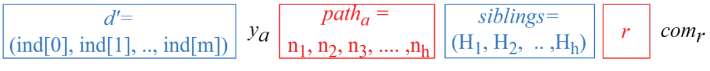

Figure 16: Private information generated for proof usage.

Case 1 Proof verification passed: The order is output y_a, com_r, input data d includes 7 features, and the proof generation date.

Case 2 Tampering with π : After the proof is generated, private information is embedded within π. If the contents of π are altered, using the verification key to validate the proof will indicate that the proof has been tampered with.

Case 3 Tampering with input data d : When d is altered, by comparing the private information d' with the modified d and their corresponding outputs y_a, we can determine if the data has been unauthorizedly modified. If d fails to find the correct path to produce output y_a, existing proofs indicate that the input data has been tampered with and can no longer be considered reliable.

Case 4 Tampering with output y_a : When output y_a is tampered with, the system needs to verify whether the modified output can still find a legitimate path through the corresponding d and private information d'. If it cannot match, it proves that the output y_a has been altered, thereby affecting the system's trustworthiness and effectiveness.

Case 5 Tampering with commitment com_r : Using the data integrity verification mechanism provided by the Merkle Tree structure, by comparing the hash values of nodes and their sibling nodes with the root com_r. If these values do not match, it indicates that the commitment com_r. This suggests that the current model is not the one originally possessed.

Case 6 Tampering with keys or model in the device: If the *GenProof* function, proving key, or verification key are compromised, their interconnected design prevents the original proving key from generating valid proofs with a different function. Using a mismatched proving key requires the corresponding verification key; otherwise, verification fails.

5 Discussion about Practical Deployment

The proposed zk-Risk Engine enables trust assessment even without ground-truth labels: K-means offers an initial label-free separation of interaction records (over 95% agreement with reference labels in Case 1), and a lightweight Decision Tree converts the pseudo-labels into an interpretable, low-compute classifier that remains effective under different malicious-traffic ratios. Beyond accuracy, integrity and privacy are enhanced by embedding inference into a ZKDT workflow (Merkle-authenticated paths) and generating succinct Groth16 proofs, allowing devices to verify trust decisions without exposing model internals or enabling tampering. Nevertheless, deployment faces key challenges. IoT traffic is high-volume and non-stationary; evolving devices, firmware, and attacker strategies can cause concept drift that invalidates fixed centroids and tree thresholds, requiring drift monitoring and periodic (or event-triggered) recalibration and retraining. In addition, the ZKDT layer adds runtime cost, so proof generation/verification latency, memory footprint, and energy use must be validated on realistic embedded platforms. Finally, communication overhead for proof exchange and key distribution must be minimized and robust to intermittent connectivity, motivating secure over-the-air (OTA) updates to continuously refresh models and zk artifacts.

ZKPs offer robust verification of a decision tree's integrity, ensuring the authenticity of the model while bolstering the protocol's resilience against model tampering and corruption, including attacks like model cloning. However, ZKPs alone are not sufficient to mitigate privacy-related risks during training, data exploitation (such as membership inference), or data tampering at the source. To address these challenges, ZKPs can be integrated with established privacy-preserving techniques like Differential Privacy and Federated Learning. This combination enables the development of a comprehensive, end-to-end privacy-preserving system for IoT, enhancing both data security and model integrity across the entire lifecycle.

6 Conclusion

In conclusion, this paper proposes a machine learning-based dynamic trust assessment mechanism for IoT devices operating without explicit data labels. Using the K-means algorithm, we distinguished between normal and potentially anomalous traffic, achieving an accuracy rate of 95%. Subsequently, we conducted a more in-depth behavioral analysis using classification algorithms, employing decision trees in the label analysis after K-means clustering, simulating real-world environments with varying proportions of malicious attacks and yielding nearly 98% accuracy in trust model construction, confirming its effectiveness in label analysis following K-means clustering.

Furthermore, we also utilized the sliding window method to test the optimal amount of traffic history for real-time evaluation under different proportions of malicious attacks. We found that using 5 data points as the basis for judgment resulted in the least decrease in the model's detection rate across various malicious attack ratios. Additionally, we discovered that excessive data might weaken the model's ability to accurately detect real threats.

Additionally, to enhance device security, we have introduced the ZTA and integrated ZKP technology into the evaluation mechanism. By eliminating any trust relationships between devices and utilizing ZKP, we achieve secure collaboration and data exchange without exposing internal mechanisms. This ensures the integrity of trust evaluations while protecting the IoT network from potential attackers.

Future research will focus on refining risk classification by introducing more granular risk levels and expanding the feature set to reduce false positives. We will also develop a dynamic risk engine selection process that integrates the outputs of multiple specialized risk engines to provide comprehensive risk management.

Acknowledgement: This research was supported by the National Science and Technology Council (NSTC), Taiwan.

Funding Statement: This work received funding support from the National Science and Technology Council (NSTC), Taiwan, under Grant Nos. 114-2410-H-011-026-MY3; 114-2221-E-011-115; 114-2634-F-011-002-MBK.

Author Contributions: Study supervision, methodological discussions, research analysis, and manuscript revision, Nai-Wei Lo; Manuscript drafting, sponsorship acquisition, confirmation of research methods, and provision and analysis of experimental data, Cheng-I Lin; Study supervision, methodological discussions, manuscript revision, the initial and final adjustment of the manuscript content, Chih-Chieh Chang; Experiments and research methodologies, Chi-Yang Chang; Experimental adjustment, refinement of research methods, and final verification of the manuscript content, Tran Thi Luu Ly. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: No datasets were generated or analyzed during the current study.

Ethics Approval: This paper does not contain any studies with human participants or animals performed by any of the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Aaqib M, Ali A, Chen L, Nibouche O. IoT trust and reputation: a survey and taxonomy. J Cloud Comput. 2023;12(1):42. doi:10.1186/s13677-023-00416-8.
2. Sivapriyan R, Sushmitha SV, Pooja K, Sakshi N. Analysis of security challenges and issues in IoT enabled smart homes. In: 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS); 2021 Dec 16–18; Bangalore, India. p. 1–6. doi:10.1109/CSITSS54238.2021.9683324.

3. Siwakoti YR, Bhurtel M, Rawat DB, Oest A, Johnson RC. Advances in IoT security: vulnerabilities, enabled criminal services, attacks, and countermeasures. IEEE Internet Things J. 2023;10(13):11224–39. doi:10.1109/jiot.2023.3252594.
4. Karie NM, Sahri NM, Yang W, Valli C, Kebande VR. A review of security standards and frameworks for IoT-based smart environments. IEEE Access. 2021;9:121975–95. doi:10.1109/access.2021.3109886.
5. Iqbal W, Abbas H, Daneshmand M, Rauf B, Bangash YA. An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security. IEEE Internet Things J. 2020;7(10):10250–76. doi:10.1109/jiot.2020.2997651.
6. Awadelkarim Mohamed AM, Abdallah M, Hamad YM. IoT security: review and future directions for protection models. In: 2020 International Conference on Computing and Information Technology (ICCIT-1441); 2020 Sep 9–10; Tabuk, Saudi Arabia. p. 1–4. doi:10.1109/ICCIT-144147971.2020.9213715.
7. Rose S, Borchert O, Mitchell S, Connelly S. Zero trust architecture (NIST Special Publication 800-207). Gaithersburg, MD, USA: National Institute of Standards and Technology; 2020. doi:10.6028/NIST.SP.800-207.
8. Alhandi SA, Kamaludin H, Alduais NAM. Trust evaluation model in IoT environment: a comprehensive survey. IEEE Access. 2023;11(1):11165–82. doi:10.1109/ACCESS.2023.3240990.
9. Wang X, Garg S, Lin H, Kaddoum G, Hu J, Hassan MM. Heterogeneous blockchain and AI-driven hierarchical trust evaluation for 5G-enabled intelligent transportation systems. IEEE Trans Intell Transp Syst. 2023;24(2):2074–83. doi:10.1109/TITS.2021.3129417.
10. Nurse JRC, Creese S, De Roure D. Security risk assessment in internet of things systems. IT Prof. 2017;19(5):20–6. doi:10.1109/mitp.2017.3680959.
11. Yassine I, Halabi T, Bellaiche M. Security risk assessment methodologies in the internet of things: survey and taxonomy. In: 2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C); 2021 Dec 6–10; Hainan, China. p. 668–75. doi:10.1109/qrs-c55045.2021.00101.
12. Mishina R, Tanimoto S, Goromaru H, Sato H, Kanai A. Risk management of silent cyber risks in consideration of emerging risks. In: 2021 10th International Congress on Advanced Applied Informatics (IIAI-AAI); 2021 Jul 11–16; Niigata, Japan. p. 710–6. doi:10.1109/iiai-aai53430.2021.00126.
13. Griffioen P, Sinopoli B. Assessing risks and modeling threats in the internet of things. arXiv:2110.07771. 2021.
14. Alwahedi F, Aldhaheri A, Ferrag MA, Battah A, Tihanyi N. Machine learning techniques for IoT security: current research and future vision with generative AI and large language models. Inter Things Cyber-Phys Syst. 2024;4(4):167–85. doi:10.1016/j.iotcps.2023.12.003.
15. Adam M, Hammoudeh M, Alrawashdeh R, Alsulaimy B. A survey on security, privacy, trust, and architectural challenges in IoT Systems. IEEE Access. 2024;12(4):57128–49. doi:10.1109/ACCESS.2024.3382709.
16. Leka E, Hoxha E, Rexha G. Security and privacy concerns associated with the internet of things (IoT) and the role of adapting blockchain and machine learning: a systematic literature review. In: 2023 46th MIPRO ICT and Electronics Convention (MIPRO); 2023 May 22–26; Opatija, Croatia. p. 1690–6. doi:10.23919/MIPRO57284.2023.10159869.
17. Jayasinghe U, Lee GM, Um T-W, Shi Q. Machine learning based trust computational model for IoT services. IEEE Trans Sustain Comput. 2018;4(1):39–52. doi:10.1109/tsusc.2018.2839623.
18. Yang L, Li Y, Yang SX, Lu Y, Guo T, Yu K. Generative adversarial learning for intelligent trust management in 6G wireless networks. IEEE Netw. 2022;36(4):134–40. doi:10.1109/mnet.003.2100672.
19. Jiang J, Zhu X, Han G, Guizani M, Shu L. A dynamic trust evaluation and update mechanism based on C4.5 decision tree in underwater wireless sensor networks. IEEE Trans Veh Technol. 2020;69(8):9031–40. doi:10.1109/tvt.2020.2999566.
20. Han G, He Y, Jiang J, Wang N, Guizani M, Ansere JA. A synergetic trust model based on SVM in underwater acoustic sensor networks. IEEE Trans Veh Technol. 2019;68(11):11239–47. doi:10.1109/tvt.2019.2939179.
21. Ali-Eldin AMT. A hybrid trust computing approach for IoT using social similarity and machine learning. PLoS One. 2022;17(7):e0265658. doi:10.1371/journal.pone.0265658.

22. Sagar S, Mahmood A, Wang K, Sheng QZ, Pabani JK, Zhang WE. Trust-SIoT: toward trustworthy object classification in the social internet of things. IEEE Trans Netw Serv Manag. 2023;20(2):1210–23. doi:10.1109/tnsm.2023.3247831.
23. Si X, Yuan S, Jiang K, Fan Z. Research on access control model of zero trust based on clustering algorithm. In: 2022 3rd International Conference on Electronics, Communications and Information Technology (CECIT); 2022 Dec 23–25; Sanya, China. p. 397–401. doi:10.1109/CECIT58139.2022.00075.
24. Zhang J, Fang Z, Zhang Y, Song D. Zero knowledge proofs for decision tree predictions and accuracy. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20). New York, NY, USA: ACM; 2020. p. 2039–53. doi:10.1145/3372297.3417278.
25. Liu T, Xie X, Zhang Y. zkCNN: zero knowledge proofs for convolutional neural network predictions and accuracy. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21). New York, NY, USA: ACM; 2021. p. 2968–85. doi:10.1145/3460120.3485379.
26. Singh N, Dayama P, Pandit V. Zero knowledge proofs towards verifiable decentralized AI pipelines. In: Eyal I, Garay J, editors. Financial cryptography and data security. Cham, Switzerland: Springer International Publishing; 2022. p. 248–75. doi:10.1007/978-3-031-18283-9_12.
27. Ghodsi Z, Gu T, Garg S. SafetyNets: verifiable execution of deep neural networks on an untrusted cloud. In: Advances in neural information processing systems 30 (NeurIPS 2017). Red Hook, NY, USA: Curran Associates, Inc.; 2017. doi:10.5555/3294996.3295220.
28. Huang C, Wang J, Chen H, Si S, Huang Z, Xiao J. zkMLaaS: a verifiable scheme for machine learning as a service. In: GLOBECOM 2022—2022 IEEE Global Communications Conference; 2022 Dec 4–8; Rio de Janeiro, Brazil. 2022. p. 5475–80. doi:10.1109/GLOBECOM48099.2022.10000784.
29. iden3. Circom Documentation; 2024 [cited 2025 Jun 13]. Available from: https://docs.circom.io/.
30. Groth J. On the size of pairing-based non-interactive arguments. In: Fischlin M, Coron J-S, editors. Advances in cryptology–EUROCRYPT 2016. Berlin/Heidelberg, Germany: Springer; 2016. p. 305–26. doi:10.1007/978-3-662-49896-5_11.
31. Neto ECP, Dadkhah S, Ferreira R, Zohourian A, Lu R, Ghorbani AA. CICIoT2023: a real-time dataset and benchmark for large-scale attacks in IoT environment. Sensors. 2023;23(13):5941. doi:10.3390/s23135941.