Future Technology
Open Access Journal
ISSN 2832-0379

November 2026| Volume 05 | Issue 04 | Pages 257-269

Journal homepage: https://fupubco.com/futech

https://doi.org/10.55670/fpll.futech.5.4.21

Future Publishing LLC

Article

Probabilistic-temporal modeling of network security monitoring subprocesses for multi-stage attacks in IoT networks

Ammar Dawood Jasim[1]*, Mokhalad Al-Tameemi[2]

[1]College of Information Engineering, Al-Nahrain University, Jadriya, Baghdad, Iraq
[2]Department of Sociology, College of Arts, Wasit University, Al-Kut, Wasit, Iraq

ARTICLE INFO

Article history:
Received 22 March 2026
Received in revised form
29 August 2026
Accepted 18 September 2026

Keywords:
Internet of Things, Network security monitoring, Recurrent neural networks, Multi-stage attacks, Anomaly detection

*Corresponding author
Email address:
ammar.alaythawy@nahrainuniv.edu.iq

DOI: 10.55670/fpll.futech.5.4.21

ABSTRACT

Timely detection of a multistage cyberattack in an IoT network depends on the chosen monitoring architecture, execution time, and the transition behavior of subprocesses within that architecture. This paper introduces a probabilistic temporal model to evaluate an NSMS for both known and unknown multi-stage attacks. First, we introduce a conceptual monitoring architecture that organizes streaming-data processing, temporal analysis, attack classification, attack prediction, and response selection. The recurrent neural networks in the architecture form the basis of the implementation framework and are neither trained nor empirically analyzed. The evaluated contribution represents the NSMS subprocesses as a continuous-time Markov chain, where each state corresponds to a monitoring task and each transition rate is estimated from the respective mean processing time. We develop state-probability equations and use a specialized software tool to estimate the stationary probabilities and successful anomaly-detection probabilities. Numerical evaluations examine how delays in preprocessing, syntactic construction, encoding, classification, and prediction affect monitoring effectiveness. The analysis shows that the estimated probability of successful anomaly detection decreases as preprocessing and encoding time increases, underscoring the need to account for computational delay in time-critical security monitoring. The sensitivity and steady-state analyses identify the subprocesses that most affect the modeled system's behavior. The developed framework enables quantitative evaluation of NSMS timing constraints and allocation of computational resources before designing and experimentally evaluating the learning-based architecture.

1. Introduction

The rapid growth of modern Data Transmission Networks (DTNs), especially within Internet of Things (IoT) ecosystems, has increased the number of interlinked devices, communication channels, and diverse data flows. While it offers greater potential for automation and information transfer, it also creates a larger attack surface that malicious actors can exploit. In these conditions, cyber-attacks are not restricted to direct or isolated intrusion attempts. In contrast, many cyber-attacks use multi-stage attacks (MSAs), where multiple logically connected stages are executed in sequence to gain unauthorized access while remaining hidden [1-4]. Unlike one-shot attacks, MSAs are hard to detect because their actions are divided across several stages and can appear as legitimate network activity when each stage is analyzed separately. The standard MSA life cycle includes reconnaissance, in which attackers gather information on the target network using open-source intelligence, scanning, and port probes. It is followed by tool preparation and attack planning, where attackers select appropriate exploits, malware, or attack frameworks based on the information they gathered about vulnerabilities. Next, attackers intrude into the target network by exploiting vulnerabilities or gaining unauthorized access. The next step involves privilege escalation and persistence, where attackers increase their control over the compromised network and create backdoors or hidden services. Lastly, trace covering is performed by encrypting network traffic, using proxies, or modifying logs to reduce the chances of detection [5-7]. As shown in Figure 1, the attack lifecycle in IoT DTNs consists of several attack stages, observable security indicators, and associated monitoring processes. These stages produce different indicators, including unusual scans, suspicious tools, unsuccessful logins, unidentified processes, service changes, anomalous resource access, and invisible traces. These indicators are significant information for the Network

Security Monitoring System (NSMS) and are analyzed through traffic monitoring, anomaly detection, attack classification, attack prediction, and countermeasures. The link between attack progression and the security monitor's response lies at the heart of the proposed monitoring framework. Intelligent security monitors that provide active protection against MSAs must operate across multiple attack stages. Moreover, they must not only identify individual anomalies but also determine the temporal correlation between network activity and the attack stage, and predict future attack behavior. This is especially necessary in IoT infrastructure because connected devices differ in computational power, communication protocols, security settings, and traffic patterns [8-12].

Previous work can be classified into four mutually supportive categories. Multistage attacks are detected using recurrence and sequence-to-sequence models that account for interdependencies among attack stages [13,14]. Researchers have also explored gated recurrence models such as LSTM and GRU for temporal data processing [15]. Studies comparing the effectiveness of time-series forecasting using LSTMs have recently shown their capability to represent non-linear dependencies; nevertheless, their application to the cybersecurity domain needs to be proven separately [16].

Traditional CNN-RNN intrusion detection systems are usually trained offline on labeled benchmarks, while online self-supervised methods learn on incoming traffic without an offline training step [17,18]. Edge-IIoTset is a heterogeneous IoT/IIoT traffic generator for IDS benchmarking experiments [19]. Risk probabilistic models estimate security risks from system metadata, but they do not consider subprocess monitoring times [20].

Table 1 compares multi-stage, recurrent, streaming, and probabilistic security-monitoring algorithms based on methodological differences, dataset considerations, attack models, and evaluation criteria. This table further shows that the current research focuses on CTMC-based analysis of monitoring subprocess times, not on packet-level detection accuracy. The main contribution of this study is the formulation of a probabilistic-temporal framework for studying the internal subprocesses of an NSMS during attacks at various stages, whether known or unknown. The developed RNN-based blocks offer a conceptual design framework for future temporal attack classification and prediction frameworks, but their learning performance is not evaluated here. The proposed model uses continuous-time Markov states and intensities to model the effects of processing delay on the stationary probability of detection success [21,22].

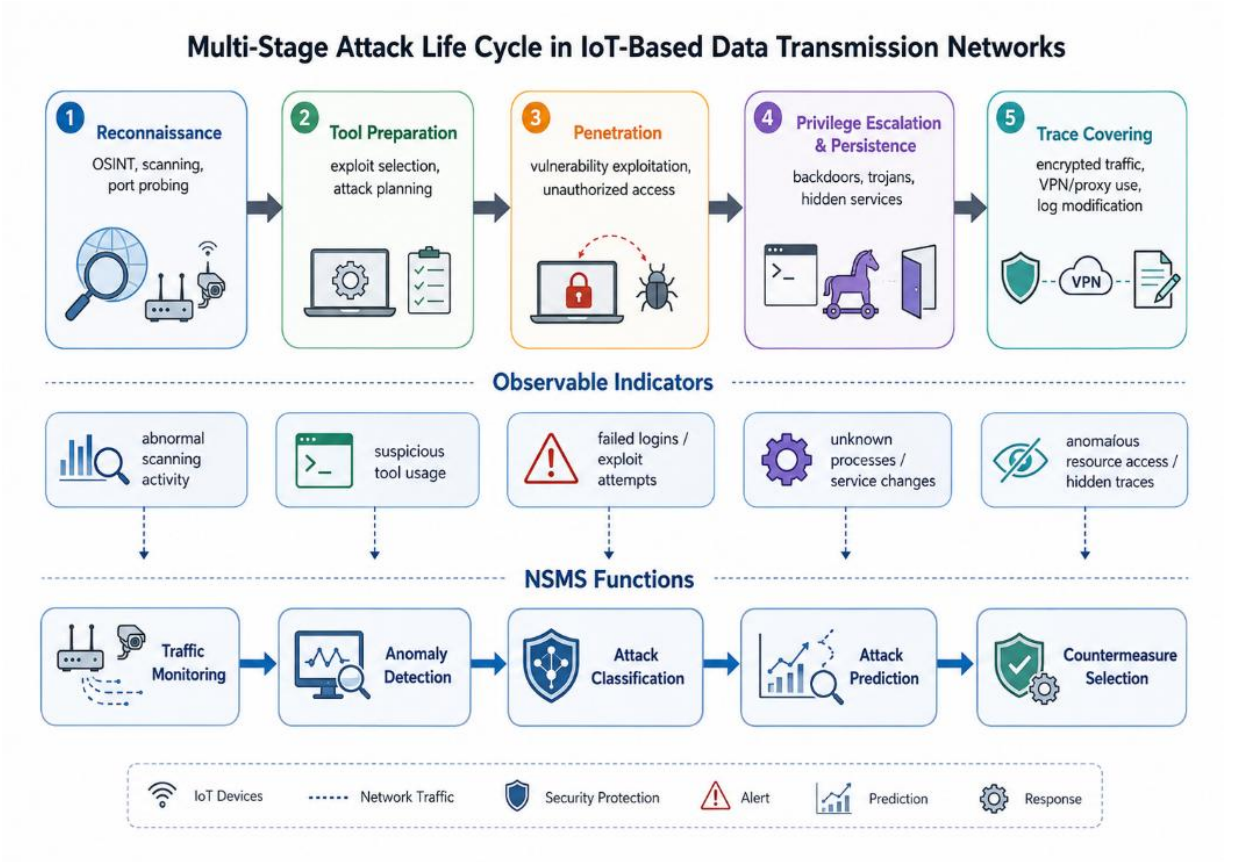


Figure 1. Multi-stage attack monitoring in IoT networks

Table 1. Comparison of related intrusion-detection and probabilistic-monitoring studies

Study	Method and processing mode	Data	Attack model	Reported metrics	Relation to the present study
Li et al. [13]	B-MLSTM; offline training with adaptive retraining	CTU-13, Gas-Water, and AWID	Low-frequency and multi-stage IIoT attacks	False-positive and false-negative rates	Present study analyzes monitoring-subprocess timing using a CTMC
Zhou et al. [14]	LSTM encoder-decoder seq2seq; batch learning	Alert sequences from four public testbed datasets	Ordered multi-stage attacks	F1 score and stage-detection performance	Present study evaluates process delays and stationary state probabilities
Weerakody et al. [15]	Review of LSTM and GRU models	Irregular time-series studies	Not attack-specific	Study-dependent prediction metrics	Provides background for selecting recurrent architectures
Khan et al. [17]	Hybrid CNN–RNN; offline supervised learning	CSE-CIC-IDS2018	Individual network-attack classes	Accuracy, precision, recall, and F1 score	Present study evaluates process timing rather than classifier accuracy
Nakip and Gelenbe [18]	Self-supervised AADRNN; online streaming	Kitsune and Bot-IoT	Botnet traffic and compromised devices	Accuracy, balanced accuracy, TPR, and TNR	Provides online adaptation, whereas the present model is analytical
Ferrag et al. [19]	IoT/IIoT dataset with ML/DL benchmarks	Edge-IIoTset	Fourteen IoT/IIoT attack types	Accuracy, precision, recall, and F1 score	Provides a suitable dataset for future experimental validation
Palko et al. [20]	Decision-tree-based risk assessment	Security metadata and system metrics	Cyber-risk classification	Classification effectiveness	Present study explicitly models monitoring-state transitions and delays

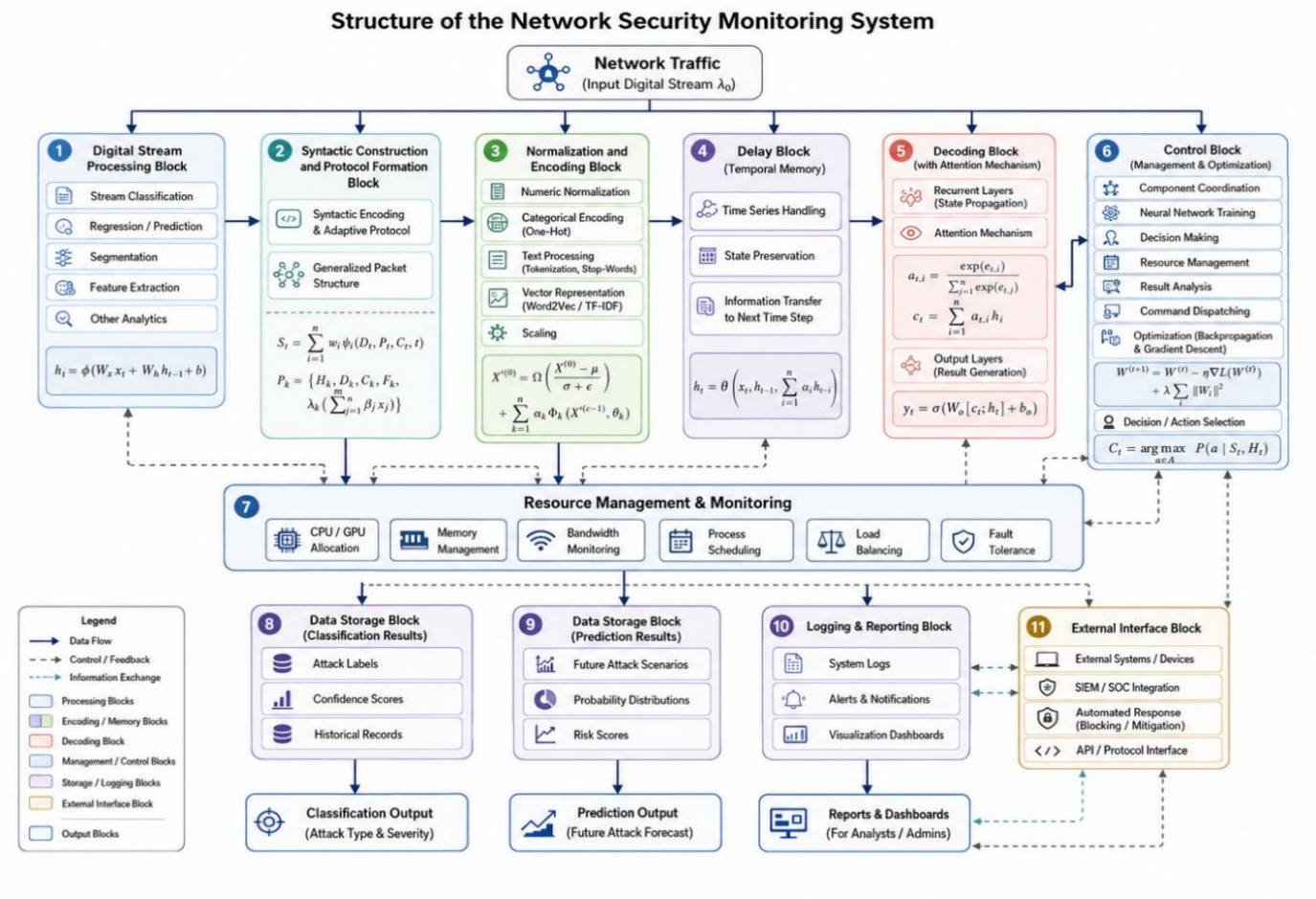


Figure 2. Conceptual functional architecture of the proposed NSMS

2. Conceptual architecture of the network security monitoring system

The architecture that is described in this section defines the operational environment needed to build the probabilistic-temporal model proposed in Section 3. While the recurrent neural network, attention mechanism, self-organizing map, tree-based classification, and attack prediction elements of this architecture define a conceptual implementation paradigm and not an experimentally proven one, none of the neural network components have been trained or tested on any dataset of network traffic; thus, no conclusions are drawn about their classification performance, prediction performance, generalization capabilities, or superior performance compared to other machine learning approaches. To address the challenges of detecting and predicting multistage attacks in Internet of Things-based data transmission networks, this paper proposes an NSMS architecture based on these concepts, shown in Figure 2. The architecture groups key monitoring subprocesses, including data gathering from the network, protocol parsing, feature preprocessing, temporal buffering, recurrent processing, output decoding, system control, and decision-making support. These functional blocks form the foundation of the probabilistic-temporal model designed in Section 3. However, the current work neither trains nor experimentally tests the neural components. Consequently, the equations that are introduced in this section constitute the interface of the future neural implementation and do not claim to be any novel learning algorithm. The neural components featured in the architectural model make use of existing recurrent-processing equations. Equation (1) refers to the traditional recurrent hidden-state updating equation, Equations (8)-(9) constitute one such attention equation, and Equations (22)-(23) refer to contextual feedback, which is compatible with a Jordan recurrent neural network.

The current paper does not propose or test a separate synapse control signal, adaptive synaptic rule, gating process, or fixed two-layer sequential arrangement in experimentation. Therefore, these neural mechanisms are not innovative methodological contributions and are included solely to illustrate computational implementations of the subprocesses outlined in the probabilistic-temporal model. Figure 3 illustrates the end-to-end data flow of the revised conceptual NSMS architecture. During each processing cycle, the system acquires heterogeneous traffic, protocol, device, and historical-security attributes as a unified data record. These data undergo deterministic validation, encoding, scaling, and feature integration to produ, an element of double-struck cap R to the d vector $x_t \in \mathbb{R}^d$. The most recent L feature vectors are then arranged as the temporal input matrix $X_t \in \mathbb{R}^{L\times d}$ and passed to the single recurrent encoder. The encoder generates the hidden-state matrix $H_t \in \mathbb{R}^{L\times h}$, while the optional attention mechanism summarizes the relevant temporal information in the context vector $c_t \in \mathbb{R}^h$.

Utilizing these representations, the decoder estimates the anomaly probability p_t^{anom}, attack-class probability vector $p_t^{\text{cls}} \in \mathbb{R}^C$, and next-stage probability vector $p_t^{\text{next}} \in \mathbb{R}^M$. After processing cycle t, the knowledge base is given a single structured entry that includes the timestamp, device/session ID, the anomaly probability, probability vectors of the attack class and class, probability vectors for the next stage and the next stage, as well as the respective confidence levels. The recurrent transient states and model parameters are not included in the knowledge base in terms of storage and updates. The historical security input component of the vector during cycle t + 1 comes only from the already existing inference records. This pipeline defines the interfaces, data types, and tensor dimensions of the conceptual architecture without any redundancy. A simpler reference architecture includes an LSTM and a first-order Markov monitor.

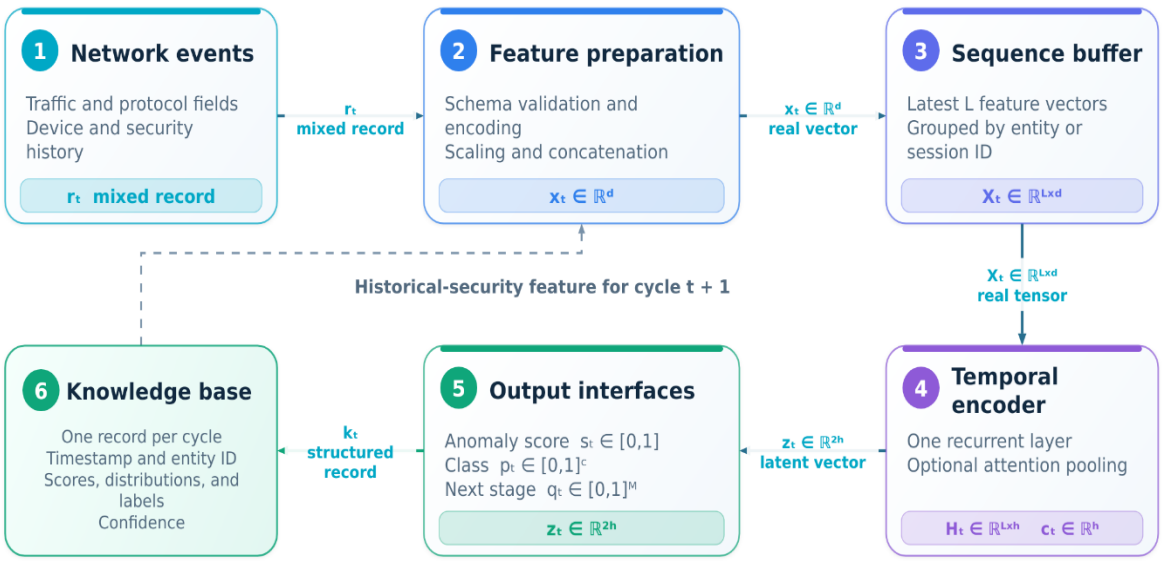


Figure 3. Conceptual functional architecture of the proposed NSMS

Both architectures rely on the same encoded feature vectors and temporal input windows from the LSTM. In the reference architecture, the LSTM predicts the current anomaly or attack state, while the Markov transition matrix predicts the next attack stage based on the current state. In the proposed NSMS, the recurrent network, attention-based context vector, and three different output heads are used for anomaly detection, attack classification, and prediction of the next attack stage, respectively. Because this study did not train or evaluate any of these neural modules, it provides no evidence of empirical superiority over the LSTM-Markov baseline. Future comparisons can include identical preprocessing, data partitions, sequence lengths, training procedures, and evaluation of classification accuracy, false alarm rate, next-stage prediction accuracy, parameter count, and inference time. An ablation study should investigate the importance of the attention mechanism in otherwise identical models. Table 2 shows the roles of the selected NSMS stages compared with the simpler baseline.

The digital stream processing component gets a steady stream of timestamped network events. Each event may include network traffic details, protocol-related details, device details, and past security details. The feature vector at time t can be defined as:

$$x_t = \left[(x_t^{\mathrm{tr}})^T \quad (x_t^{\mathrm{pr}})^T \quad (x_t^{\mathrm{dv}})^T \quad (x_t^{\mathrm{hs}})^T\right]^T \in \mathbb{R}^d \tag{1}$$

where x_t^{tr}, x_t^{pr}, x_t^{dv}, and x_t^{hs}contain the traffic, protocol, device, and historical-security features, respectively. Their corresponding dimensions are d_{tr}, d_{pr}, d_{dv}, and d_{hs}, such that $d = d_{\mathrm{tr}} + d_{\mathrm{pr}} + d_{\mathrm{dv}} + d_{\mathrm{hs}}$. Equation (1) establishes a single, consistently indexed input vector for all subsequent processing blocks.

Because multistage attacks develop through temporally correlated events, individual feature vectors cannot capture this progress. The delay block will then keep an ordered window with the latest L observations:

$$X_t = \begin{bmatrix} x_{t-L+1}^T \\ x_{t-L+2}^T \\ \vdots \\ x_t^T \end{bmatrix} \in \mathbb{R}^{L\times d} \tag{2}$$

where X_t denotes the input sequence ending at time t, L is the sequence-window length, and d is the feature dimension defined in Eq. (1). Equation (2) specifies the temporal input supplied to the recurrent-processing block. In a future experimental implementation, L must be selected according to the traffic-sampling interval and the expected duration of the attack stages.

As shown in Figure 2, the syntactic construction and protocol formation block serves as the parsing and interpretation stage for packets. It parses protocol fields, separates header and payload information, checks length and integrity fields when present, and extracts the metadata needed for security monitoring. The block neither creates nor modifies any packets being sent. It just translates the traffic flow into a common format. The normalization and encoding step prepare the extracted features for further processing. In case of the future data-driven implementation, numerical features would be normalized to comparable scales, while categorical features like the type of protocols, connection state, and IDs of the devices would be encoded appropriately. Missing or erroneous observations would also be addressed depending on the nature of the particular chosen dataset. Because the current investigation does not involve training and validating a neural network architecture on any particular dataset, no parameters for normalization, categorical encoding, feature selection, or missing-value handling were developed or investigated. This stage remains at the conceptual level, and the standard z-score and TF-IDF formulas are not provided as numbered equations. The time processing block gives a theoretical approach to recurrently extracting temporal information from the sequenced inputs X_t as illustrated in the previous Eq. (2). The conventional hidden recurrent state of the model can be given as:

$$h_i = \phi\left(W_x x_i + W_h h_{i-1} + b_h\right),\ i = t-L+1, \ldots, t \tag{3}$$

where $h_i \in \mathbb{R}^m$ is the hidden-state vector at time i, h_{i-1} is the preceding hidden state, $W_x \in \mathbb{R}^{m\times d}$ is the input-weight matrix, $W_h \in \mathbb{R}^{m\times m}$ is the recurrent-weight matrix, $b_h \in \mathbb{R}^m$ is the bias vector, $m$is the hidden-state dimension, and $\phi(\cdot)$ is a nonlinear activation function. The initial state h_{t-L} may be initialized to zero or transferred from the preceding sequence window. Equation (3) is included only to define the temporal interface of the conceptual architecture. It represents a conventional recurrent update and is not claimed as a controlled-synapse rule or a novel neural formulation.

Table 2. Functional comparison of the retained NSMS stages with the LSTM–Markov baseline

Retained stage	Relationship to the LSTM–Markov baseline	Justification
Encoding and feature integration	Used by both architectures	Converts heterogeneous input records into a consistently defined feature vector
Sequence buffer	Used by the LSTM component	Preserves the temporal order of network events
Single recurrent encoder	Can be instantiated as an LSTM	Replaces multiple overlapping neural modules with one temporal representation
Optional attention	Not required by the simpler baseline	Retained only as a testable extension whose benefit must be established through ablation
Output heads	Represent the same prediction targets	Explicitly separate anomaly detection, attack classification, and next-stage prediction
Knowledge base	Serves as the monitoring-history store	Maintains structured inference records for auditing and subsequent-cycle features rather than functioning as a trainable model

The decoding block converts the temporal representation into monitoring outputs. If the optional attention mechanism is retained in a future implementation, the relevance score associated with each hidden state is defined as:

$$e_{t,i} = v_a^T \tanh\ (W_a h_i + U_a h_t + b_a),\ i = t - L + 1, \dots, t \quad (4)$$

where $e_{t,i}$ is the unnormalized relevance score assigned to h_i, $W_a, U_a \in \mathbb{R}^{q \times m}$ are trainable attention matrices, $v_a \in \mathbb{R}^q$ is a trainable projection vector, $b_a \in \mathbb{R}^q$ is the attention bias, and q is the attention-space dimension. Equation (4) is required because the relevance score must be defined before the corresponding attention coefficient can be calculated.

The normalized attention coefficient is then obtained as:

$$\alpha_{t,i} = \frac{\exp\ (e_{t,i})}{\sum_{j=t-L+1}^{t} \exp\ (e_{t,j})},\ i = t - L + 1, \dots, t \quad (5)$$

where $\alpha_{t,i}$ is represents, the relative importance assigned to h_i. By construction, $\alpha_{t,i} > 0$ and $\sum_{i=t-L+1}^{t} \alpha_{t,i} = 1$.

The attention-weighted context vector is subsequently calculated as:

$$c_t = \sum_{i=t-L+1}^{t} \alpha_{t,i}\, h_i,\ c_t \in \mathbb{R}^m \quad (6)$$

where c_t summarizes the hidden states within the selected sequence window. Equations (5) and (6) complete the definition of the optional attention operation. These equations describe a standard attention formulation and do not constitute a methodological contribution of the present study.

The decoder input depends on whether the optional attention mechanism is retained:

$$z_t = \begin{cases} [h_t^T \quad c_t^T]^T, & \text{if attention is employed,} \\ h_t, & \text{if attention is omitted.} \end{cases} \quad (7a)$$

Thus, $z_t \in \mathbb{R}^{d_z}$, where $d_z = 2m$when attention is employed and $d_z = m$otherwise. The conceptual decoder may then generate an anomaly probability, an attack-class probability vector, and a next-stage probability vector as follows:

$$p_t^{\text{anom}} = \sigma\left(w_d^T z_t + b_d\right) \quad (7b)$$

$$p_t^{\text{cls}} = \text{softmax}\ (W_{\text{cls}} z_t + b_{\text{cls}}) \quad (7c)$$

$$p_t^{\text{next}} = \text{softmax}\ (W_{\text{next}} z_t + b_{\text{next}}) \quad (7d)$$

where the $p_t^{\text{anom}} \in [0, 1]$ is referring to the estimated anomaly probability, $\sigma(\cdot)$ is referring to the sigmoid function, $p_t^{\text{cls}} \in \mathbb{R}^C$contains the probabilities of Cattack classes, and the $p_t^{\text{next}} \in \mathbb{R}^M$contains the probabilities of the M possible subsequent attack stages. Moreover, the $w_d \in \mathbb{R}^{d_z}$, $b_d \in \mathbb{R}$, $W_{\text{cls}} \in \mathbb{R}^{C \times d_z}$, $b_{\text{cls}} \in \mathbb{R}^C$, $W_{\text{next}} \in \mathbb{R}^{M \times d_z}$, and $b_{\text{next}} \in \mathbb{R}^M$are referring to the parameters of the three output heads. It's necessary to mention that Equations (7a)–(7d) establish a mathematically complete connection between the temporal representation and the monitoring outputs shown in the conceptual architecture. They do not imply that these parameters were learned or that the corresponding outputs were experimentally generated in the present study.

For the future trained implementation, the probability outputs may be converted into a discrete monitoring decision using the following expressions:

$$\hat{y}_t^{\text{anom}} = \mathbb{I}\ (p_t^{\text{anom}} \geq \tau) \quad (8a)$$

$$\hat{C}_t = p_{t,k}^{\text{cls}} \quad (8b)$$

$$\hat{S}_{t+1} = p_{t,r}^{\text{next}} \quad (8c)$$

where $\hat{y}_t^{\text{anom}} \in \{0, 1\}$is the anomaly decision, $\mathbb{I}(\cdot)$is the indicator function, $\tau \in (0, 1)$is an anomaly-decision threshold, $\hat{C}_t$is the estimated attack class, and $\hat{S}_{t+1}$is the predicted subsequent attack stage. In a future experimental implementation, τmust be determined using validation data rather than selected arbitrarily. In the present study, these quantities remain conceptual outputs and are not used as empirical performance measures.

The control module coordinates the NSMS by managing data flow, task scheduling, resource allocation, output recording, and information exchange between functional modules. The control module manages the proper order of operations and ensures proper transfer of processed information between the analysis, classification, prediction, and decision-making modules. The recurrent processing module estimates neural parameters, while the control module orchestrates system operation and computation.

The multifunctional processing unit serves as the interface linking feature processing with attack decision-making. As shown in Figure 4, the first unit consists of three processing steps: unsupervised pattern organization, neural feature representation, and supervised classification. The self-organizing map organizes network traffic data by similarity and helps discover hidden structures, while the tree-based classifier provides an interpretable approach to classifying observations into predetermined network or attack classes. In the current modeling process, these components form the functional structure and information flow used to model specific NSMS subprocesses in the probabilistic-temporal model.

As part of this process chain, the feature integration phase merges traffic, protocol, device, and security history features into the common feature vector described by Eq. (1). This feature representation provides a consistent interface between preprocessing and temporal analysis while maintaining the distinctiveness of each feature type. Processed temporal features theoretically enable network state estimation and attack classification tasks. Correlation analysis and risk-cost assessment are implementation-specific functions of the decision support subsystem and were not examined in this study. After identifying and classifying an anomaly into an attack category, the resulting security information can be relayed to a policy-based decision support system for action selection. These actions could include filtering traffic, isolating connections, enhancing authentication, restricting services, generating alarms, and notifying administrators. Future action selection systems will need to account for attack severity, residual risk, implementation cost, and computational resources.

However, the mathematical formulation describes the feature representation, temporal processing, attention, decoding, and decision interfaces needed for NSMS to function coherently. Nonetheless, the quantitative analysis is limited to probabilistic temporal state transitions and sub-process execution time. Therefore, the presented results cannot be taken as experimental analysis of the neural network's performance. The next-step prediction is achieved using an output head designed specifically for this purpose. Consequently, the second Jordan RNN, contextual feedback network, and output aggregation formula are unnecessary. The model receives history information related to securities for the next step only from inference data saved in the knowledge base. This model gives an opportunity to implement the interface for further probabilistic-temporal analysis.

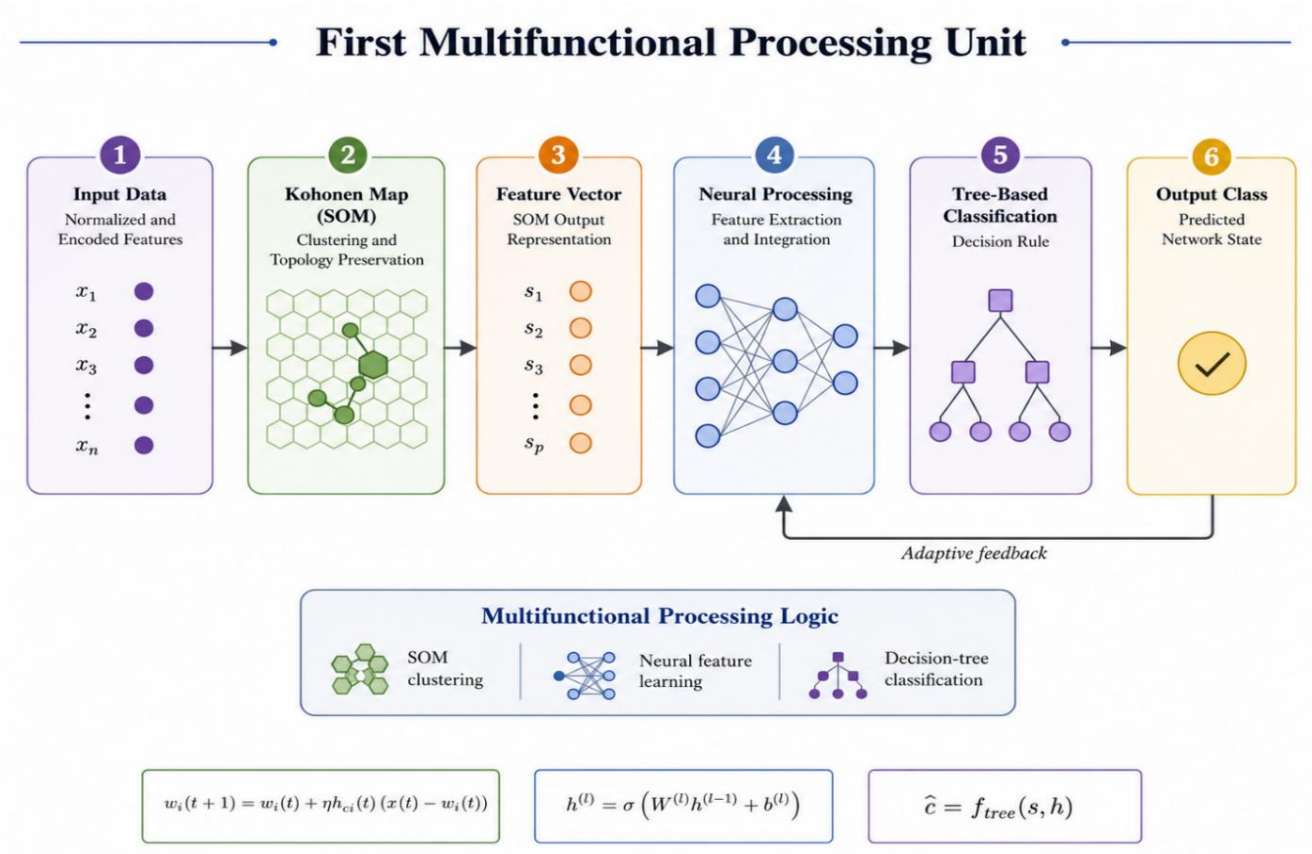


Figure 4. Conceptual structure of the first multifunctional processing unit

2.1 Deployment and resource considerations

In an effective implementation of NSMS, there is a distinction between simple tasks performed by the edge and complex processing at the monitor site. Traffic monitoring, protocol decoding, feature extraction, normalization, and buffering occur at the edge, while recurrent processing, optional attention, classification, next-stage prediction, CTMC analysis, and knowledge base management take place at the security monitor. The SOMs and tree models shown in Figure 4 are theoretical examples of what would be done at the central monitor, not on resource-limited IoT devices. MQTT, CoAP, and HTTP flows may be expressed via standard flow parameters, along with protocol-specific parameters such as message types, packet sizes, topics or resource identifiers, response codes, connection status, and interarrivals. For TLS- or DTLS-encrypted flows, we propose considering observable flow parameters, such as headers, timings, and endpoints' metadata, without assuming access to the decrypted payload. For a temporal buffer containing L records with $d$single-precision features, the approximate feature-storage requirement is $M_{\text{buffer}} \approx 4Ld$ bytes, excluding the system overhead. The online processing latency is defined as $T_{\text{online}} = T_{\text{capture}} + T_{\text{parse}} + T_{\text{encode}} + T_{\text{inference}} + T_{\text{decision}}$ and should remain below the sequence-window update interval. The reported processing times, including the 1000 srecurrent-context value, are assumed CTMC sensitivity parameters rather than measured edge-device execution times. Concept drift and countermeasure optimization are not implemented in the present model; periodic monitor-side retraining and policy-constrained response selection are reserved for future experimental work.

3. Probabilistic modeling of network security monitoring under multi-stage attacks

To accurately analyze network security monitoring in DTNs subject to MSAs, the proposed Network Security Monitoring System (NSMS) is modeled as a probabilistic system with a varying structure. This means the system's operation depends on the current operating regime, namely, the monitoring, defense, prediction, countermeasures generation, and recovery regimes. Switching from one regime to another depends on the system's current state and follows probability laws. In other words, the NSMS is a continuous-time Markov process in which each state is associated with a particular monitoring task, and each transition rate equals the average switching time between subprocesses. The general model of state transitions may be represented as follows [23]:

$$\frac{dP_i(t)}{dt} = \sum_{j=0}^{N} \lambda_{ji} P_j(t) - P_i(t) \sum_{j=0}^{N} \lambda_{ij} \quad i = 0,1,\dots,N \qquad (9)$$

where $\left(P_{i(t)}\right)$ denotes the probability that the NSMS is in the $(i) - th$ state at time (t), $(\{ji\})$ *is the transition intensity from state* $\left(S_j\right)$*to state* (S_i), $(\{ij\})$ is the transition intensity from state (S_i) to state $\left(S_j\right)$, and (N) is the total number of system states. The transition intensity is defined according to the average execution time of each subprocess as:

$$\lambda_{i\to j} = \frac{1}{\bar{t}_{i\to j}} \qquad (10)$$

Unlike conventional accuracy indicators, which describe only the correctness of attack classification, and unlike temporal indicators, which describe only detection speed, probabilistic-

temporal characteristics provide a more complete measure of the monitoring process. They allow the estimation of the probability that an intrusion or anomaly will be detected within a specified period of time. The probability of successful anomaly detection can be described by [24]:

$$P_A(t) = 1 - e^{-\lambda_d t} \quad (11)$$

where $(P_{A(t)})$ is the probability of successful anomaly detection at time (t), and (λ_d) is the effective detection rate. A higher value of (λ_d) indicates faster and more reliable detection, while a lower value indicates that the NSMS requires more time to detect attack-related events [25].

Figure 5 presents the state graph of the NSMS during network security monitoring under MSA conditions. In the graph, each state (S_i) represents a specific monitoring subprocess, while Each transition intensity λ_{ij} represents the rate at which the NSMS moves from state S_i to state S_j during the monitoring process. The main parameters used to characterize the monitoring process are summarized in Table 3. Based on the state graph, the probabilistic behavior of the NSMS is described by the following system of differential equations:

$$\frac{dP_0(t)}{dt} = \lambda_{5\to0}P_5(t) + \lambda_{6\to0}P_6(t) - \lambda_{0\to1}P_0(t) \quad (12)$$

$$\frac{dP_1(t)}{dt} = \lambda_{0\to1}P_0(t) - (\lambda_{1\to2} + \lambda_{1\to7})P_1(t) \quad (13)$$

$$\frac{dP_2(t)}{dt} = \lambda_{1\to2}P_1(t) - (\lambda_{2\to3} + \lambda_{2\to8})P_2(t) \quad (14)$$

$$\frac{dP_3(t)}{dt} = \lambda_{2\to3}P_2(t) + \lambda_{11\to3}P_{11}(t) - (\lambda_{3\to4} + \lambda_{3\to11})P_3(t) \quad (15)$$

$$\frac{dP_4(t)}{dt} = \lambda_{3\to4}P_3(t) - (\lambda_{4\to5} + \lambda_{4\to6})P_4(t) \quad (16)$$

$$\frac{dP_5(t)}{dt} = \lambda_{4\to5}P_4(t) + \lambda_{7\to5}P_7(t) - \lambda_{5\to0}P_5(t) \quad (17)$$

$$\frac{dP_6(t)}{dt} = \lambda_{4\to6}P_4(t) + \lambda_{10\to6}P_{10}(t) - \lambda_{6\to0}P_6(t) \quad (18)$$

$$\frac{dP_7(t)}{dt} = \lambda_{1\to7}P_1(t) - \lambda_{7\to5}P_7(t) \quad (19)$$

$$\frac{dP_8(t)}{dt} = \lambda_{2\to8}P_2(t) - \lambda_{8\to9}P_8(t) \quad (20)$$

$$\frac{dP_9(t)}{dt} = \lambda_{8\to9}P_8(t) + \lambda_{11\to9}P_{11}(t) - (\lambda_{9\to10} + \lambda_{9\to11})P_9(t) \quad (21)$$

$$\frac{dP_{10}(t)}{dt} = \lambda_{9\to10}P_9(t) - \lambda_{10\to6}P_{10}(t) \quad (22)$$

$$\frac{dP_{11}(t)}{dt} = \lambda_{3\to11}P_3(t) + \lambda_{9\to11}P_9(t) - (\lambda_{11\to9} + \lambda_{11\to3})P_{11}(t) \quad (23)$$

The state-probability vector is defined as:

$$p(t) = [P_0, (t), P_1, (t) \dots P_{11}(t)]^T \quad (24)$$

Accordingly, the complete system of balance equations can be written as:

$$\frac{dp(t)}{dt} = Q^T p(t) \quad (25)$$

Assuming that each monitoring cycle starts in the normal input-waiting state S_0, the initial condition is:

$$\mathrm{p}(0) = [1 \quad 0 \quad 0 \quad \cdots \quad 0]^{\mathrm{T}} \quad (26)$$

The CTMC generator matrix is denoted by $Q = [q_{ij}]_{12\times12}$, where each off-diagonal element represents a transition intensity and each diagonal element is the negative sum of the outgoing transition intensities:

$$q_{ij} = \begin{cases} \lambda_{i\to j}, & i \neq j \text{ and the transition exists,} \\ -\sum_{k\neq i}\lambda_{i\to k}, & i = j, \\ 0, & \text{otherwise.} \end{cases} \quad (27)$$

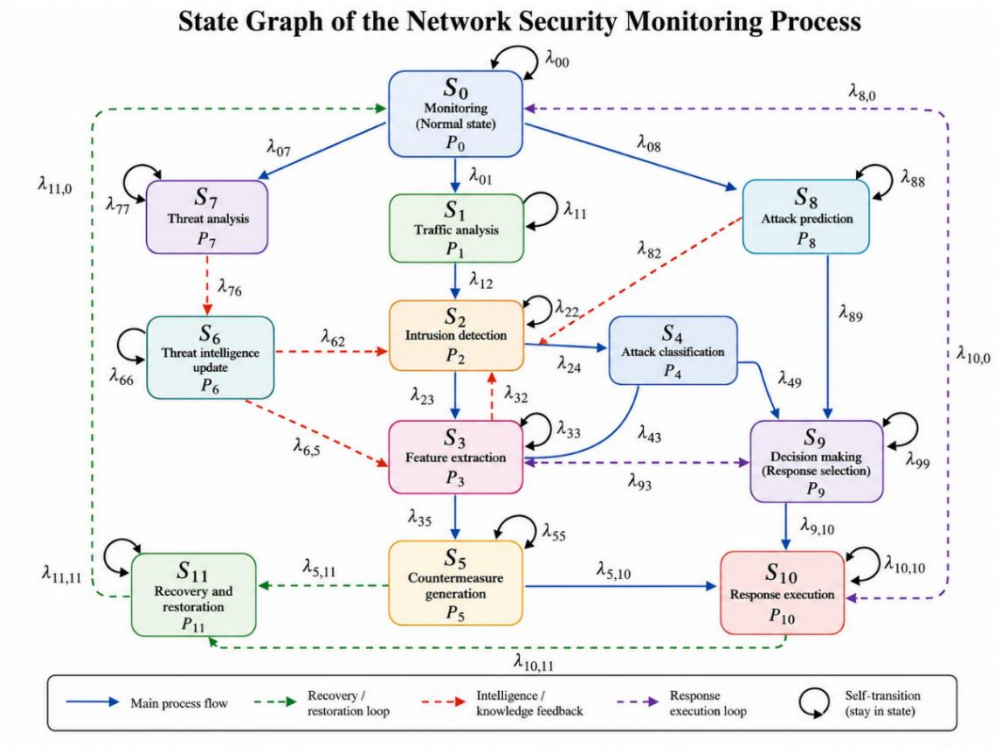


Figure 5. State graph of the NSMS under multi-stage attack conditions

Table 3. Parameters of the network security monitoring process under multi-stage attacks

State	Description of state	Transition intensity (s^{-1})	Parameter	Description of parameter	Value (s)
S_0	The NSMS waits for incoming digital-stream data before starting a monitoring cycle.	$\lambda_{0\to1} = \frac{1}{\bar{t}_0}$	$\bar{t}_0$	Mean waiting and acquisition time of the incoming digital stream	2000
S_1	The NSMS performs syntactic construction and protocol formation for the incoming data stream.	$\lambda_{1\to2} = \lambda_{1\to7} = \frac{1}{\bar{t}_{FSC}}$	$\bar{t}_{FSC}$	Mean syntactic-construction and protocol-formation time	10, 20, 30
S_2	The NSMS normalizes, encodes, and prepares the input data for recurrent processing.	$\lambda_{2\to3} = \lambda_{2\to8} = \frac{1}{\bar{t}_{OD}}$	$\bar{t}_{OD}$	Mean input-data preprocessing time	500baseline; 100–500sensitivity range
S_3	The NSMS analyzes temporal dependencies in the classification branch.	$\lambda_{3\to4} = \lambda_{3\to11} = \frac{1}{\bar{t}_{AM1}}$	$\bar{t}_{AM1}$	Mean temporal-analysis time in the classification branch	110
S_4	The NSMS generates the decoder output used in the classification branch.	$\lambda_{4\to5} = \lambda_{4\to6} = \frac{1}{\bar{t}_{G1}}$	$\bar{t}_{G1}$	Mean output-generation time in the primary processing branch	1
S_5	The NSMS classifies an unknown multistage attack and estimates its type and severity.	$\lambda_{5\to0} = \frac{1}{\bar{t}_{CK}}$	$\bar{t}_{CK}$	Mean multistage attack-classification time	100
S_6	The NSMS predicts the possible future behavior or next stage of a known attack.	$\lambda_{6\to0} = \frac{1}{\bar{t}_{CP}}$	$\bar{t}_{CP}$	Mean next-stage prediction time	300
S_7	The NSMS retrieves and transfers historical information supporting the classification branch.	$\lambda_{7\to5} = \frac{1}{\bar{t}_{CH1}}$	$\bar{t}_{CH1}$	Mean historical-information transfer time for classification	100
S_8	The NSMS retrieves and transfers historical information supporting the prediction branch.	$\lambda_{8\to9} = \frac{1}{\bar{t}_{CH2}}$	$\bar{t}_{CH2}$	Mean historical-information transfer time for prediction	120
S_9	The NSMS analyzes temporal dependencies in the prediction branch.	$\lambda_{9\to10} = \lambda_{9\to11} = \frac{1}{\bar{t}_{AM2}}$	$\bar{t}_{AM2}$	Mean temporal-analysis time in the prediction branch	115
S_{10}	The NSMS generates the decoder output used in the prediction branch.	$\lambda_{10\to6} = \frac{1}{\bar{t}_{G2}}$	$\bar{t}_{G2}$	Mean output-generation time in the secondary processing branch	130
S_{11}	The NSMS performs recurrent-context inference and temporal refinement using fixed model parameters.	$\lambda_{11\to3} = \lambda_{11\to9} = \frac{1}{\bar{t}_{OP}}$	$\bar{t}_{OP}$	Mean recurrent-context processing time	1000

The nonzero transitions are:

$E = (0,1), (1,2), (1,7), (2,3), (2,8), (3,4), (3,11), (4,5), (4,6),$

$(5,0), (6,0), (7,5), (8,9), (9,10), (9,11), (10,6), (11,3), (11,9).$

By construction, all the off-diagonal elements are nonnegative; every row of Qsums to zero, and all elements of Qhave units of s^{-1}. Every state utilized in the balance equations is listed separately. The mean processing times are expressed in seconds, whereas the corresponding transition intensities are expressed in $s^{-1}$and calculated as $\lambda_{i\to j} = 1/\bar{t}_{i\to j}$. The values 10, 20, and 30 sassigned to $\bar{t}_{FSC}$, together with the range of 100–500 sassigned to $\bar{t}_{OD}$, represent the settings used in the sensitivity analysis. The stationary probability vector was calculated by setting all state-probability derivatives to zero and solving $Q^T\pi = 0$together with the normalization condition $1^T\pi = 1$. One row of $Q^T$was replaced by the normalization equation, and the resulting linear system was solved in double precision using LU decomposition. Because the stationary system was solved algebraically, no numerical integration step size was required. The solution was accepted when all probabilities were nonnegative, $| 1^T\pi - 1 | < 10^{-10}$, and $\| Q^T\pi \|_\infty < 10^{-10}$. At steady state, the derivatives in (25)-(29) are set to zero, and the probabilities must satisfy the normalization condition:

$$\sum_{i=0}^{11} P_i = 1 \tag{28}$$

The stationary probability vector can be expressed in normalized form as [26]:

$$\mathrm{P} = [P_0, P_1, P_2, \dots, P_{11}]^T, \qquad \mathrm{Q}^T\mathrm{P} = 0, \qquad 1^T\mathrm{P} = 1 \tag{29}$$

where Q is the transition-intensity (generator) matrix derived from the state graph, Pis the stationary probability vector, 0 is the zero vector, and 1is a vector of ones used to satisfy the probability normalization constraint. To express the stationary probabilities in a compact form, the base probability is defined as:

$$P_0 = K, \qquad P_i = q_i K, \quad i = 1,2, \dots ,11 \tag{30}$$

where (q_i) represents the relative probability coefficient of state (S_i). The normalization coefficient is then given by [27]:

$$K = \frac{1}{1+\sum_{i=1}^{11} q_i} \tag{31}$$

Algorithm 1 presents the computational algorithm for stationary-state probabilities and anomaly-detection probability modeling.

Algorithm 1. Stationary-probability and detection-probability calculation

Input:

- States $S_0 - S_{11}$
- Directed transition set E
- Mean transition times $\bar{t}ij$ in seconds
- Attack scenario c ∈ {unknown, known}
- Investigated $\bar{t}OD$ and $\bar{t}FSC$ values

Output:

- Stationary probabilities $\pi_0 - \pi_{11}$
- Detection probability P_A
- Normalization error and residual norm

Procedure:

1. For each investigated parameter combination:
2. Initialize Qas a 12 × 12 zero matrix.
3. For every transition $i \to j$ in E:
   - Calculate $\lambda_i j = 1/\bar{t}ij$.
   - Assign $Q_{ij} = \lambda_{ij}$.
4. For each state i, calculate:
   - Qii = −Σj≠i Qij.
5. Set $A = Q^{\mathrm{T}}$ and initialize b as a 12-element zero vector.
6. Replace the last row of A with [1,1, … ,1] and set $b_{12} = 1$.
7. Solve $A(\pi = b)$using double-precision LU decomposition.
8. Calculate:
   - Normalization error = $\Sigma\pi_i - 1$
   - Residual norm = $||Q^{\mathrm{T}}\pi||\infty$
9. If the scenario is unknown:
   - $P_A = \pi_0 + \pi_5$.
10. Otherwise, if the scenario is known:

- $P_A = \pi_0 + \pi_6$.

11. Store $\bar{t}OD$, $\bar{t}FSC$, π, PA, normalization error, and residual norm.
12. Repeat until all parameter combinations have been evaluated.
13. Export the stored results for plotting.

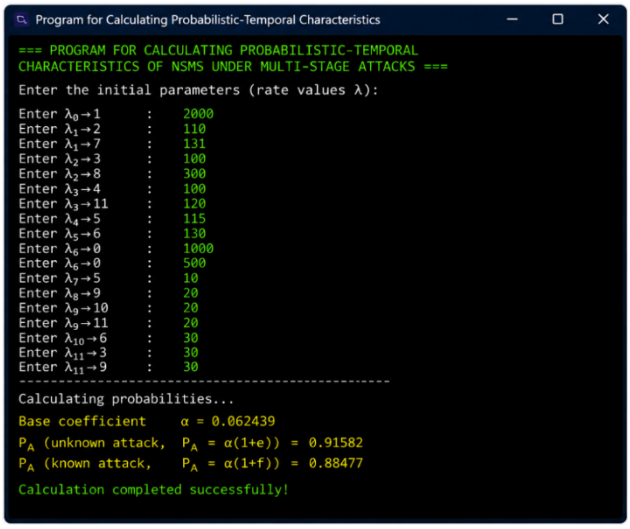
The algorithm computes the generator matrix using the transition intensities and then solves the normalized stationary equation to find cap P sub cap A for each case considered.

4. Numerical Study of the Monitoring-Process Model

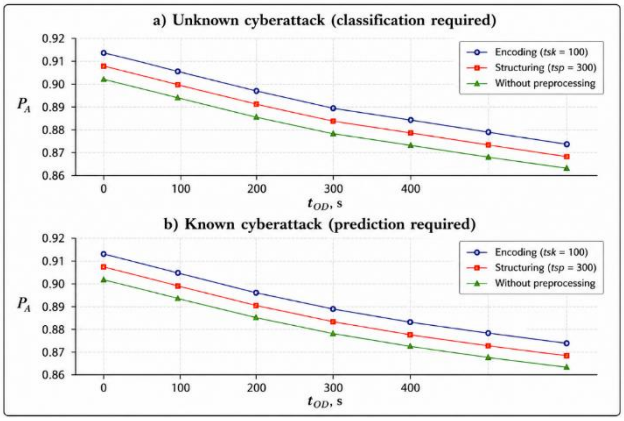
Based on the system of differential equations and the set of parameters that have been given in Table 2, the probabilistic temporal model was tested using a special software realization of the model. The created software accepts the necessary state-transition parameters via the console, calculates the stationary state probabilities according to the developed mathematical model, and produces automatically generated graphs of results for performance testing. As can be seen from Figure 6(a), the console allows for entering the state-transition parameters and performing the calculations of the probabilities. The corresponding graph of the results is given in Figure 6(b), where the dependence of the probability of anomaly detection (P_{a}) on the input data processing time (t_{o}ɒ) is demonstrated for the cases of the unknown and known multi-stage cyberattacks. The obtained results describe the relationship between the probability of successful anomaly detection (P_A) and the processing time of input data before transfer to the recurrent neural network, denoted by t_{OD}. Two operating cases are considered. In the first case, an unknown cyberattack is detected, and further classification is required. In the second case, a known cyberattack is detected, and prediction of the possible attack scenario is required. The corresponding probability curves are shown in Figure 7.

The numerical study considers the effects of processing delay on the steady-state behavior of the probabilistic-temporal monitoring model. The model-defined anomaly-detection probability, P_A, decreases as the mean preprocessing time, $\bar{t}_{\mathrm{OD}}$, increases, with a similar trend observed for the syntactic-construction time, $\bar{t}_{\mathrm{FSC}}$. Longer processing intervals reduce the corresponding transition intensities according to $\lambda_{i\to j} = b_{ij}/\bar{t}_i$and increase the residence time in the associated states. Parameter variability was examined deterministically by varying $\bar{t}_{\mathrm{OD}}$and $\bar{t}_{\mathrm{FSC}}$while holding the remaining parameters at their baseline values. Thus, the curves obtained show only sensitivity tendencies rather than any confidence intervals. The experiment analyzes the behavior of a modeled process and does not include any tests for actual packet detection precision or generalizability. This requires separate validation using labeled multi-stage IoT traffic data and measuring detection precision, false positive ratio, and processing time.

To further examine the steady-state response and sensitivity of the probabilistic–temporal model to processing delays, two additional analyses were conducted. Figure 8 presents the stationary probabilities of states S_0–S_{11} using the baseline parameter settings listed in Table 3. Figure 9 illustrates how the successful anomaly-detection probability, P_A, changes with the mean input-data preprocessing time, $\bar{t}_{\mathrm{OD}}$, for $\bar{t}_{\mathrm{FSC}}$values of 10, 20, and 30 s. These findings reflect the calculated behavior for the probabilistic–temporal model and should not be regarded as experimental validation of the conceptual neural components.



(a)



(b)

Figure 6. Software interface for calculating probabilistic-temporal characteristics of the NSMS (a) Software interface for NSMS probability calculation (b) Probability of successful anomaly detection under different attack scenarios

Using the baseline settings $\bar{t}_{\mathrm{FSC}} = 20$ sand $\bar{t}_{\mathrm{OD}} = 500$ s, the stationary probability vector was obtained by solving $Q^T\pi = 0$, subject to $1^T\pi = 1$. The resulting

$$\pi = [0.732903, , 0.003665, , 0.045806, , 0.010077, , 0.000046, , 0.022903, \; 0.041226\; 0.018323\; 0.010994\; 0.010535\; 0.011910\; 0.091613]^T$$

All probabilities were nonnegative. Using the full-precision values, their sum was 1.000000, and the residual norm was $\|Q^T\pi\|_\infty = 9.91 \times 10^{-20}$, confirming the numerical consistency of the stationary solution. The displayed values sum to 1.000001because they are rounded to six decimal places. The previous Figure 8 shows that state S_0 has the highest stationary probability, $P_0 = 0.732903$, indicating that the NSMS spends approximately 73.29%of its long-run operating time waiting for and acquiring incoming data. This dominance is consistent with the comparatively long time assigned to this state. State S_{11}has the second-highest probability, $P_{11} = 0.091613$, followed by $S_2$and S_6, with probabilities of 0.045806and 0.041226, respectively. In contrast, S_4has the lowest occupancy, $P_4 = 0.000046$, because output generation is assigned a much shorter processing time. These probabilities represent the long-run state occupancy of the CTMC model and should not be interpreted as classification accuracy or experimentally measured neural-network performance.

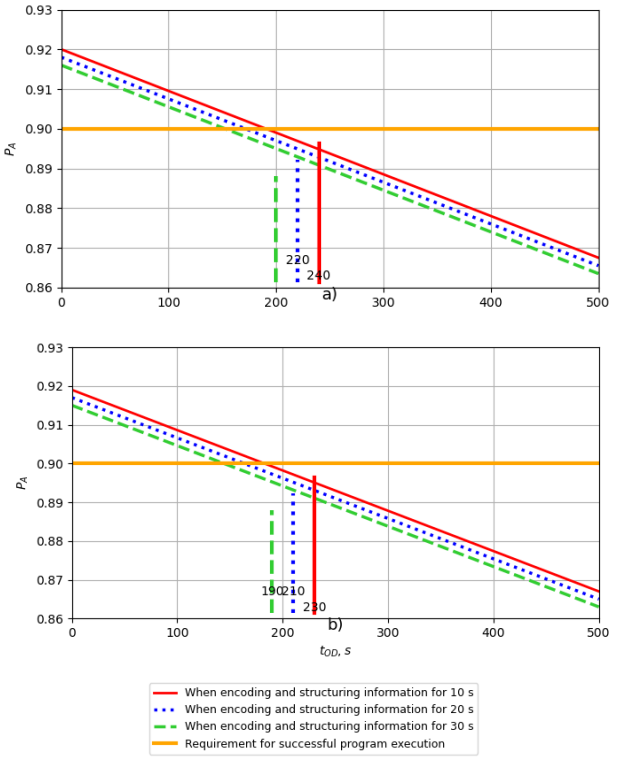


Figure 7. Probability of successful anomaly detection under different processing times (a) unknown MSA requiring classification (b) known MSA requiring prediction

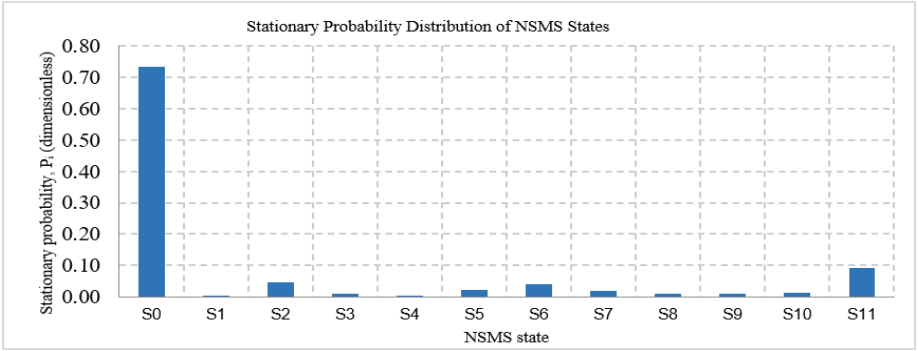


Figure 8. Stationary probabilities, P_i, of the NSMS states S_0–S_{11} (dimensionless)

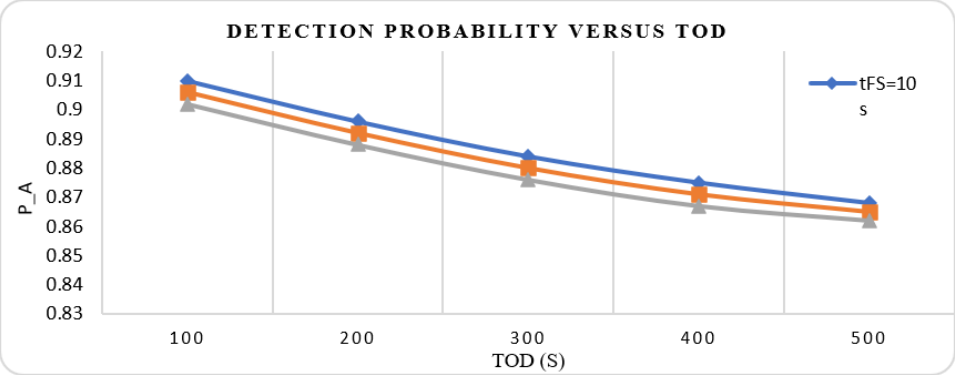


Figure 9. Stationary probabilities, P_i, of the NSMS states S_0–S_{11} (dimensionless)

5. Conclusion

This paper proposed a probabilistic-temporal modeling approach for assessing the operation of NSMS functioning under multi-staged attacks within IoT networks. The functional architecture proposed in this paper provides the organization of the major subprocesses related to the functioning of the NSMS, such as data preprocessing, temporal analysis, attack identification, attack prediction, and decision making. Nevertheless, the recurrent neural network elements are provided just as an example of implementation architecture. The contribution evaluated in this paper is the continuous-time Markov model where the major NSMS subprocesses are considered to be the states and their interaction is specified by the transition intensities. A software solution has been designed to solve the stationary-state equations and analyze the effect of the subprocesses execution time variation on the probability of anomaly detection obtained from the model. The numerical results indicate that the increase in the execution time of preprocessing, encoding, and syntactic construction decreases the probability of successful anomaly detection. It has been also shown that the known attack scenario and the unknown attack scenario have different behavior from the probabilistic point of view as they take different prediction and classification routes in the state model. In addition, the sensitivity and steady-state analysis have allowed us to identify the transitions and operational states that have the strongest impact on the operation of the modeled monitoring process. The results obtained can be used analytically to set the timing requirements, detect potential computational bottlenecks, and allocate resources for developing an NSMS for IoT systems. However, the results presented should not be viewed as experimental results of neural network classification or prediction. The current work is therefore restricted to analysis and numerical evaluation of the proposed probabilistic-temporal NSMS model, since the neural part of the proposed conceptual framework has not been trained or tested based on network traffic data. Further research will focus on implementing and evaluating the neural components using a multi-stage or advanced persistent threat sequence dataset, possibly in combination with existing IoT intrusion datasets, e.g., CIC-IoT2023, Edge-IIoTset, or ToN-IoT. The experimental design will define the input features, sequence windowing method, chronological train-test split, neural network architecture, synapse control, optimizer, loss function, early stopping, and class imbalance solution.

The performance of the implemented model will be assessed with respect to precision, recall, F1 score, false alarm rate, detection latency, and next stage prediction accuracy. The implemented model will be also benchmarked against LSTM, GRU, BiLSTM, CNN-RNN, and other state-of-the-art sequence-based multi-stage attack detection methods to evaluate the possible benefits of the proposed neural architecture.

Ethical issue

The authors are aware of and comply with best practices in publication ethics, specifically regarding authorship (avoidance of guest authorship), dual submission, manipulation of figures, competing interests, and compliance with research ethics policies. The authors adhere to publication requirements that the submitted work is original and has not been published elsewhere. All survey participants provided informed consent before participating, and the anonymity and confidentiality of all respondents were strictly maintained throughout the research process.

Data availability statement

The manuscript contains all the data. However, additional data will be provided by the corresponding author upon reasonable request.

Conflict of interest

The authors declare no potential conflict of interest.

References

[1] N. Kalpani, N. Rodrigo, D. Seneviratne, et al., “Resilient cybersecurity: Ensemble deep learning and reinforcement learning for Next-Gen IDS,” Iran Journal of Computer Science, vol. 9, Art. no. 19, 2026, doi: 10.1007/s42044-025-00364-3.

[2] K. Zhao, Z. Zhang, Z. Liang, Z. Zhang and G. Quan, "Bi-NLSTM: A Multi-Stage Attack Detection Approach for Industrial Internet," in Chinese Journal of Electronics, vol. 35, no. 2, pp. 713-724, March 2026, doi: 10.23919/cje.2025.00.254.

[3] X. Li, M. Xu, P. Vijayakumar, N. Kumar, and X. Liu, “Detection of low-frequency and multi-stage attacks in industrial Internet of Things,” IEEE Transactions on Vehicular Technology, vol. 69, no. 8, pp. 8820–8831, 2020, doi: 10.1109/TVT.2020.2995133.

[4] A. Pinto, L. C. Herrera, Y. Donoso, and J. A. Gutierrez, “Survey on intrusion detection systems based on

machine learning techniques for the protection of critical infrastructure," Sensors, vol. 23, no. 5, Art. no. 2415, 2023, doi: 10.3390/s23052415.

[5] P. Zhou, G. Zhou, D. Wu, and M. Fei, "Detecting multi-stage attacks using sequence-to-sequence model," Computers & Security, vol. 105, Art. no. 102203, 2021, doi: 10.1016/j.cose.2021.102203.

[6] S. Mishra, W. B. Alotaibi, M. Alshehri, and S. Saxena, "Cyber-attacks visualisation and prediction in complex multi-stage network," International Journal of Computer Applications in Technology, vol. 68, no. 4, pp. 345–356, 2022, doi: 10.1504/IJCAT.2022.125180.

[7] P. B. Weerakody, K. W. Wong, G. Wang, and W. Ela, "A review of irregular time series data handling with gated recurrent neural networks," Neurocomputing, vol. 441, pp. 161–178, 2021, doi: 10.1016/j.neucom.2021.02.046.

[8] M. A. Khan, "HCRNNIDS: Hybrid convolutional recurrent neural network-based network intrusion detection system," Processes, vol. 9, no. 5, Art. no. 834, 2021, doi: 10.3390/pr9050834.

[9] I. Al-Turaiki and N. Altwaijry, "A convolutional neural network for improved anomaly-based network intrusion detection," Big Data, vol. 9, no. 3, pp. 233–252, 2021, doi: 10.1089/big.2020.0263.

[10] H. Gao, "Design of network data information security monitoring system based on big data technology," Procedia Computer Science, vol. 228, pp. 348–355, 2023, doi: 10.1016/j.procs.2023.11.040.

[11] D. Olszewski, "A data-scattering-preserving adaptive self-organizing map," Engineering Applications of Artificial Intelligence, vol. 105, Art. no. 104420, 2021, doi: 10.1016/j.engappai.2021.104420.

[12] H. Yang, X. Li, W. Qiang, Y. Zhao, W. Zhang, and C. Tang, "A network traffic forecasting method based on SA optimized ARIMA–BP neural network," Computer Networks, vol. 193, Art. no. 108102, 2021, doi: 10.1016/j.comnet.2021.108102.

[13] X. Li, M. Xu, P. Vijayakumar, N. Kumar, and X. Liu, "Detection of low-frequency and multi-stage attacks in Industrial Internet of Things," IEEE Trans. Veh. Technol., vol. 69, no. 8, pp. 8820–8831, 2020, doi: 10.1109/TVT.2020.2995133.

[14] P. Zhou, G. Zhou, D. Wu, and M. Fei, "Detecting multi-stage attacks using sequence-to-sequence model," Comput. Secur., vol. 105, Art. no. 102203, 2021, doi: 10.1016/j.cose.2021.102203.

[15] P. B. Weerakody, K. W. Wong, G. Wang, and W. Ela, "A review of irregular time series data handling with gated recurrent neural networks," Neurocomputing, vol. 441, pp. 161–178, 2021, doi: 10.1016/j.neucom.2021.02.046.

[16] N. Basil, N. Cherif, M. B. Alhamadani, H. Hasan, and T. F. Agajie, "Comparative analysis of LSTM, HHOSOA, COAESPSO, and XGBoost models for time-series load forecasting," Int. J. Comput. Eng. Artif. Intell., vol. 1, no. 1, Jun. 2026. Accessed: Sep. 13, 2026. https://ijceai.org/index.php/ijceai/article/view/13.

[17] M. A. Khan, "HCRNNIDS: Hybrid convolutional recurrent neural network-based network intrusion detection system," Processes, vol. 9, no. 5, Art. no. 834, 2021, doi: 10.3390/pr9050834.

[18] M. Nakıp and E. Gelenbe, "Online self-supervised deep learning for intrusion detection systems," IEEE Trans. Inf. Forensics Security, vol. 19, pp. 5668–5683, 2024, doi: 10.1109/TIFS.2024.3402148.

[19] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications," IEEE Access, vol. 10, pp. 40281–40306, 2022, doi: 10.1109/ACCESS.2022.3165809.

[20] D. Palko et al., "Cyber security risk modeling in distributed information systems," Appl. Sci., vol. 13, no. 4, Art. no. 2393, 2023, doi: 10.3390/app13042393.

[21] E. Gelenbe and M. Nasereddin, "Adaptive Attack Mitigation for IoV Flood Attacks," in IEEE Internet of Things Journal, vol. 12, no. 5, pp. 4701-4714, 1 March1, 2025, doi: 10.1109/JIOT.2025.3529615.

[22] R. F. Abbas and S. H. Mohammed, "Packet tracer-based configuration and functional simulation of a campus VoIP communication system using Cisco Call Manager Express," International Journal of Smart Control and Electrical Engineering Systems, vol. 1, no. 1, 2026. https://www.ijscees.org/index.php/pub/article/view/7.

[23] D. Palko et al., "Cyber security risk modeling in distributed information systems," Applied Sciences, vol. 13, no. 4, Art. no. 2393, 2023, doi: 10.3390/app13042393.

[24] X. Li, M. Xu, P. Vijayakumar, N. Kumar, and X. Liu, "Detection of low-frequency and multi-stage attacks in industrial Internet of Things," IEEE Transactions on Vehicular Technology, vol. 69, no. 8, pp. 8820–8831, 2020, doi: 10.1109/TVT.2020.2995133.

[25] G. A. Pavliotis, Stochastic Processes and Applications: Diffusion Processes, the Fokker–Planck and Langevin Equations. New York, NY, USA: Springer, 2014, doi:10.1007/978-1-4939-1323-7.

[26] M. Bampatsikos, I. Politis, T. Ioannidis and C. Xenakis, "Trust Score Prediction and Management in IoT Ecosystems Using Markov Chains and MADM Techniques," in IEEE Transactions on Consumer Electronics, vol. 71, no. 1, pp. 862-882, Feb. 2025, doi: 10.1109/TCE.2025.3531045.

[27] P. Zhou, G. Zhou, D. Wu, and M. Fei, "Detecting multi-stage attacks using sequence-to-sequence model," Computers & Security, vol. 105, Art. no. 102203, 2021, doi: 10.1016/j.cose.2021.102203.