

When Majority Rule Fails: Pheromone-Guided Trust Adaptation for Byzantine-Resilient Intrusion Detection

Bishal Shome¹

¹Department of Computer Science and Engineering, Sikkim Manipal Institute of Technology, Sikkim
Manipal University, Majitar, Sikkim, 737136, India

Abstract - The edge layer of IoT networks aggregates and relays sensitive traffic under tight resource constraints, making it an attractive target whose compromise can undermine the wider IoT infrastructure. N-Version Programming (NVP) is a well-established software fault-tolerance strategy in which multiple independently implemented program variants execute in parallel and their outputs are resolved through voting. Conventional NVP deployments rely on Simple Majority Voting (SMV), which assigns equal, static trust to every variant - a brittle assumption once a variant begins to fail under Byzantine fault conditions. This paper proposes ACO-Voter, an adaptive trust-weighting mechanism for NVP-based Intrusion Detection Systems (IDS) at the IoT edge. Three heterogeneous classifiers - Random Forest, Linear Support Vector Classifier, and Multilayer Perceptron - are each assigned a pheromone trust value that is reinforced on agreement with the ensemble consensus and evaporated on disagreement, enabling autonomous fault isolation without prior knowledge of which variant will fail. Evaluated on the DNN-EdgeIoT dataset under a permanent coordinated Byzantine fault scenario, ACO-Voter isolates the faulty variant within 95 packets and improves attack recall from 0.464 (SMV) to 0.634 - a 36.6% relative gain. The approach is well suited to fault-tolerant intrusion detection in IoT edge gateways, vehicular networks, and industrial IoT systems.

Key Words: Intrusion Detection System, N-Version Programming, Ant Colony Optimization, Byzantine Fault Tolerance, Adaptive Voting, IoT Edge Security.

1. INTRODUCTION

The proliferation of Internet of Things (IoT) devices in smart infrastructure, healthcare, and industrial sectors has considerably expanded the attack surface available to adversaries. IoT systems are typically composed of heterogeneous, resource-constrained endpoints operating in physically exposed environments and communicating over protocols not originally designed with security as a primary concern [1]. The edge layer, positioned between IoT endpoints and cloud infrastructure, aggregates and relays sensitive data under these constraints, making it a critical attack surface whose compromise can undermine the wider IoT ecosystem.

Machine learning-based Intrusion Detection Systems (IDS) achieve high classification accuracy on benchmark datasets; however, single-model classifiers represent a fundamental single point of failure. Under adversarial manipulation or hardware degradation, their performance can deteriorate sharply. N-Version Programming (NVP) mitigates this risk by resolving the outputs of multiple independently implemented software variants through a voting mechanism, so that the failure of a single variant does not cascade into system-wide failure [3]. The dominant voting strategy in NVP deployments, Simple Majority Voting (SMV), assigns equal and static trust to every variant throughout system operation. This static assumption means SMV cannot adapt at runtime: even once a variant begins to produce consistently incorrect outputs - whether through adversarial exploitation or hardware wear - SMV continues to weight it identically to honest variants.

This paper proposes ACO-Voter, which applies Ant Colony Optimization (ACO) principles to adaptively manage per-variant trust within NVP-based IDS deployments at the IoT edge. Each classifier variant is assigned a pheromone trust value that is reinforced when it agrees with the ensemble consensus and evaporated when it disagrees, enabling the voter to autonomously isolate a faulty variant without any prior knowledge of which variant is likely to fail.

1.1 Problem Statement

Existing NVP-based IDS deployments exhibit two related limitations. First, no runtime mechanism exists to assess the trustworthiness of individual NVP variants during operation, so a variant that begins producing systematically incorrect outputs continues to participate in the final decision on an equal footing. Second, SMV-based approaches [7] assign static, equal weights to all variants regardless of runtime behaviour, meaning a permanently degraded variant contributes to the output with the same influence as a fully honest one.

1.2 Contributions

This paper makes the following contributions:

- A pheromone-based adaptive trust-weighting mechanism, ACO-Voter, is proposed for NVP-based IDS, in which each variant's influence on the final decision is continuously adjusted through pheromone deposition and evaporation.
- A coordinated Byzantine fault model is formulated in which a subset of NVP variants is degraded with variant-specific probability and severity, simulating a realistic adversarial permanent-fault scenario at the IoT edge.
- ACO-Voter is evaluated on the DNN-EdgeIoT dataset [2] and shown to isolate the faulty variant within 95 packets of fault onset and to improve attack recall by 36.6% relative to SMV under this fault condition.
- This work forms the basis of an extended study, in preparation, that evaluates ACO-Voter under intermittent and recovery-type Byzantine fault conditions, together with formal cross-validated statistical significance testing and a sensitivity analysis of the pheromone evaporation rate.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the proposed ACO-

Voter methodology. Section 4 reports experimental results. Section 5 concludes the paper.

2. RELATED WORK

2.1 NVP-Based Intrusion Detection

N-Version Programming was introduced as a software fault-tolerance technique in which N functionally equivalent programs are developed independently and executed concurrently, with a voting algorithm resolving their outputs [3]; the reliability benefit rests on the assumption of independent faults across variants [4]. Mulkamala et al. [5] showed that combining Support Vector Machines and neural networks in an IDS achieved higher detection rates than individual classifiers on the KDD Cup 1999 dataset, and Tsai et al. [6] surveyed machine-learning IDS methods, emphasizing classifier diversity as a key determinant of multi-classifier performance. Gu and Lu [7] developed an NVP-based IDS on CICIDS2017 using heterogeneous base classifiers and reported improved detection even under artificially introduced classifier degradation. None of these approaches, however, assess the runtime trustworthiness of individual NVP variants or provide a mechanism for fault isolation.

2.2 Byzantine Fault Tolerance and Adaptive Voting

Byzantine fault tolerance was formalized by Lamport et al. [8], who showed that a system of N components can withstand up to $\lfloor (N-1)/3 \rfloor$ Byzantine failures while remaining correct; Castro and Liskov [9] later demonstrated efficient Byzantine consensus in practical distributed systems. Aung and Zaw [10] examined Byzantine fault tolerance specifically in IDS voting and showed that static majority voting is compromised once multiple versions are corrupted in a coordinated manner. On the adaptive-voting side, Kuncheva [11] demonstrated that adaptive combination procedures outperform fixed fusion strategies under non-stationary data, and the Weighted Majority Algorithm [12] provides regret guarantees through multiplicative weight updates. These multiplicative schemes, however, tie a variant's recovery speed to its current weight level, so a variant that has been heavily penalized recovers only slowly even after it resumes correct behaviour - a property this paper's pheromone deposit-evaporation rule is explicitly designed to avoid.

2.3 Ant Colony Optimization in Network Security

ACO, introduced by Dorigo and Gambardella [13], mimics ant foraging behaviour through pheromone deposition and evaporation. Prior security research has applied ACO chiefly to IDS optimization rather than to the voting mechanism itself: Abraham et al. [14] proposed an ACO-based feature-selection technique to improve IDS performance. No prior work, to the authors' knowledge, applies ACO pheromone dynamics to the voting mechanism within an NVP-based IDS, nor addresses Byzantine fault isolation through this mechanism - the gap this paper addresses.

3. PROPOSED METHODOLOGY

3.1 NVP-IDS Architecture and Initialization

Consider an IoT edge gateway that receives a stream of network packets $P = \{p_1, \dots, p_n\}$, where each packet p_i is represented by a feature vector $x_i \in \mathbb{R}^d$ and associated with a ground-truth binary label $y_i \in \{0, 1\}$ (0: normal, 1: attack). The proposed NVP-IDS comprises three independently implemented classifier variants:

$$V \rightarrow \{V1, V2, V3\} \quad (1)$$

where $V1$ is a Random Forest classifier, $V2$ is a Linear Support Vector Classifier, and $V3$ is a Multilayer Perceptron. Each variant follows a distinct algorithmic paradigm - tree-based ensemble, margin maximization, and gradient-descent neural optimization respectively - satisfying the NVP requirement of implementation independence. Three variants also satisfy the minimum $N \geq 3$ required to tolerate a single Byzantine-faulty component, consistent with the bound in [8].

3.2 Byzantine Fault Model

A Byzantine fault is a condition in which a variant produces consistently incorrect probability scores for a subset of packets. This study models a coordinated multi-variant permanent fault: for attack packets ($y_i = 1$), the corrupted score is

$$s' = s - \alpha s \quad (2)$$

where s is the original score, s' the corrupted score, and $\alpha \in [0, 1]$ the fault severity. $V1$ is faulted with probability 1.0 and severity $\alpha_1 = 0.85$; $V2$ with probability 0.40 and severity $\alpha_2 = 0.60$; $V3$ with probability 0.30 and severity $\alpha_3 = 0.50$. Gaussian noise $\varepsilon \sim N(0, \sigma^2)$ is additionally applied to all outputs ($\sigma = 0.12$ for normal packets, $\sigma = 0.20$ for attack packets) to model realistic measurement uncertainty.

3.3 Probability Score Extraction

ACO-Voter operates on continuous probability scores rather than binary predictions, since weighted voting over binary outputs is mathematically equivalent to SMV regardless of the assigned weights. The Random Forest variant produces posterior probabilities via vote-fraction aggregation across its trees; the Linear SVC variant converts its decision margin to a probability via Platt scaling,

$$\hat{p}(y=1|x) = 1 / (1 + \exp(-f(x))) \quad (3)$$

and the MLP variant outputs probabilities directly through its softmax layer.

3.4 Pheromone Initialization and Update Rule

ACO-Voter maintains a pheromone vector $\tau = (\tau_1, \tau_2, \tau_3)$, $\tau_i \in [\tau_{min}, \tau_{max}] = [0.01, 1.0]$, initialized to $\tau_0 = 1.0$ for all variants. At each timestep t , the decision is produced by pheromone-weighted aggregation:

$$d(t) = \text{threshold}(\sum_i \tau_i(t) s_i(t) / \sum_i \tau_i(t)) \quad (4)$$

Pheromone weights are then updated as

$$\tau_i(t+1) = (1-\rho)\tau_i(t) + Q \cdot J_i(t) \quad (5)$$

where $\rho \in (0, 1)$ is the evaporation rate, $Q > 0$ the deposit quantity, and $J_i(t) = 1$ if V_i 's hard prediction agrees with $d(t)$, else 0. The evaporation term $(1-\rho)$ applies regardless of agreement, imposing a natural trust ceiling, while the additive deposit $+Q$ is independent of the current pheromone level - this is what enables weight-independent recovery. For a consistently agreeing variant, the pheromone converges to

$$\tau^* = Q/\rho = 0.2/0.3 \approx 0.667 \quad (6)$$

An isolation threshold $\theta = 0.30$ is set above τ_{min} and below τ^* , and pheromone values are clipped for stability:

$$\tau_i(t+1) = \text{clip}((1-\rho)\tau_i(t) + Q \cdot J_i(t), \tau_{min}, \tau_{max}) \quad (7)$$

Algorithm 1 summarizes the procedure. Parameter values used in this study ($\rho = 0.30$, $Q = 0.20$, $\theta = 0.30$) follow the operating point identified in preliminary tuning; a full sensitivity analysis of ρ is reserved for the extended study.

Algorithm 1: ACO-Voter

```

Input: Packet stream P; trained variants {V1, V2, V3}
Output: Predictions; pheromone trajectories; isolation index
1: Initialize  $\tau \leftarrow [1.0, 1.0, 1.0]$ ;  $\tau_{min} \leftarrow 0.01$ ;  $\tau_{max} \leftarrow 1.0$ ;  $p \leftarrow 0.30$ ;  $Q \leftarrow 0.20$ ;  $\theta \leftarrow 0.30$ 
2: for  $t = 1$  to  $|P|$  do
3:   Extract scores  $s = [s_1(t), s_2(t), s_3(t)]$  from V1, V2, V3
4:   Apply Byzantine fault model if  $y_i = 1$ 
5:   Compute  $d(t) = \text{threshold}(\sum_i \tau_i s_i / \sum_i \tau_i)$ 
6:    $J_i(t) = 1$  if  $\text{round}(s_i) = d(t)$ , else  $0$ 
7:    $\tau_i(t+1) = \text{clip}((1-p)\tau_i(t) + Q \cdot J_i(t), \tau_{min}, \tau_{max})$ 
8:   Log decision, pheromone values, isolation status ( $\tau_1 < \theta$ )
9: end for
10: return predictions, pheromone trajectories, convergence index

```

Algorithm 1: Pseudocode for the proposed pheromone-based ACO-Voter update procedure.

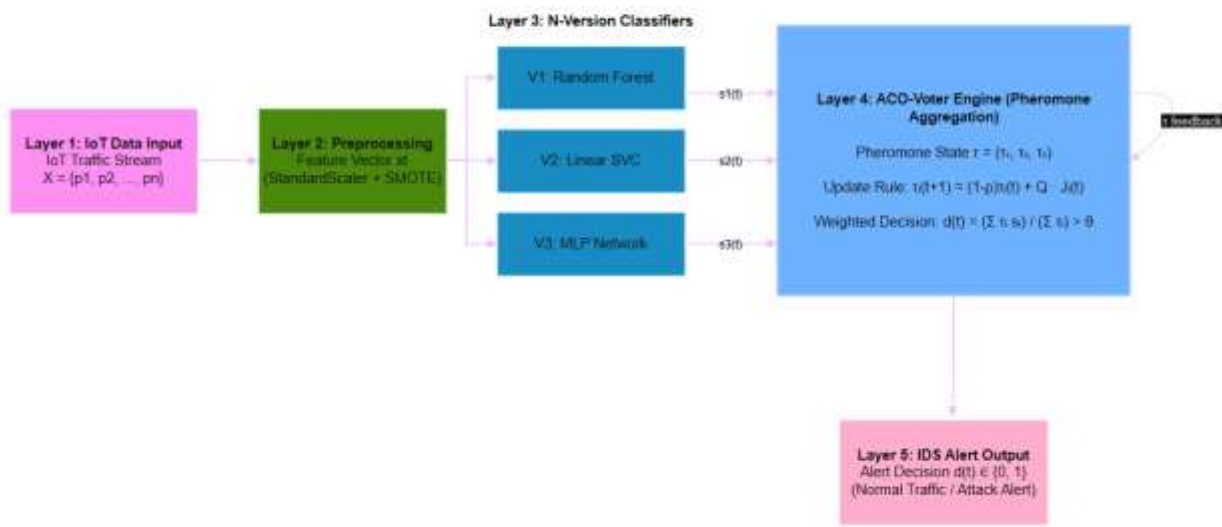


Fig. 1: System architecture of the proposed ACO-Voter framework for fault-tolerant intrusion detection at the IoT edge layer.

3.5 Architecture Workflow

Fig. 1 illustrates the resulting five-layer pipeline: (i) raw IoT traffic ingestion, (ii) feature preprocessing (StandardScaler and SMOTE), (iii) the three NVP classifier variants, (iv) the ACO-Voter pheromone-aggregation engine, and (v) the final IDS alert decision. The pheromone state feeds back into the aggregation engine at every timestep, closing the adaptive loop.

4. EXPERIMENTAL SETUP AND RESULTS

4.1 Dataset and Simulation Parameters

Experiments use a stratified sample of 100,000 packets from the DNN-EdgeIoT dataset [2], preserving the original class distribution (72.6% normal, 27.4% attack across fourteen attack categories including DDoS, SQL injection, backdoor installation, ransomware, and man-in-the-middle attacks). After removing three highly correlated ($r > 0.98$) and four zero-variance features, 51 numerical features remain. Data is split 75:25 using stratified sampling, and SMOTE with Gaussian jitter ($\sigma = 0.01$) is applied to the minority attack class in the training set only. Experiments were run in Python (scikit-learn, NumPy, SciPy) on an Intel Core i5/i7 system at 3.20 GHz with 16 GB RAM. Table 1 lists key simulation parameters.

Table 1: Simulation Parameters

Component	Parameter	Value
Random Forest (V1)	n_estimators / max_depth	100 / 15
Linear SVC (V2)	C / max_iter	1.0 / 3000
MLP (V3)	hidden layers / activation	(64,32,16) / ReLU
ACO-Voter	$p / Q / \theta$	0.30 / 0.20 / 0.30
Byzantine fault	prob. / severity (V1,V2,V3)	1.0/.85, .40/.60, .30/.50

4.2 Evaluation Metrics

Attack recall ($R = TP/(TP+FN)$) is treated as the primary metric, since missing an attack is categorically more costly than a false alarm in security-critical intrusion detection; F1-score (the harmonic mean of precision and recall) is reported alongside to capture the precision-recall trade-off, and accuracy is reported for completeness.

4.3 Results and Discussion

Table 2 compares ACO-Voter against SMV, Fixed-Weighted Voting (FWV - an oracle baseline whose weights are pre-configured using prior knowledge of which variant is most vulnerable), and an Exponential-Moving-Average (EMA) voter, under the permanent coordinated Byzantine fault scenario described in Section 3.2.

Table 2: Voter Performance under Permanent Byzantine Fault (Scenario A)

Metric	SMV	FWV	EMA	ACO
Recall	0.464	0.634	0.464	0.634
F1-score	0.621	0.758	0.621	0.768
Accuracy	0.846	0.901	0.846	0.854

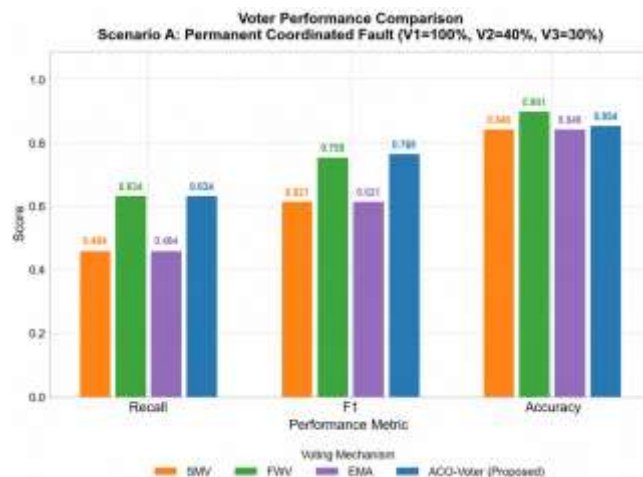


Fig. 2: Scenario A voter performance comparison (SMV, FWV, EMA, ACO) under permanent fault injection with ambient noise.

ACO-Voter improves attack recall from 0.464 (SMV) to 0.634 - a 36.6% relative improvement, corresponding to 170 additional attacks detected per 1,000 attack packets. This improvement is attributable entirely to the pheromone mechanism.

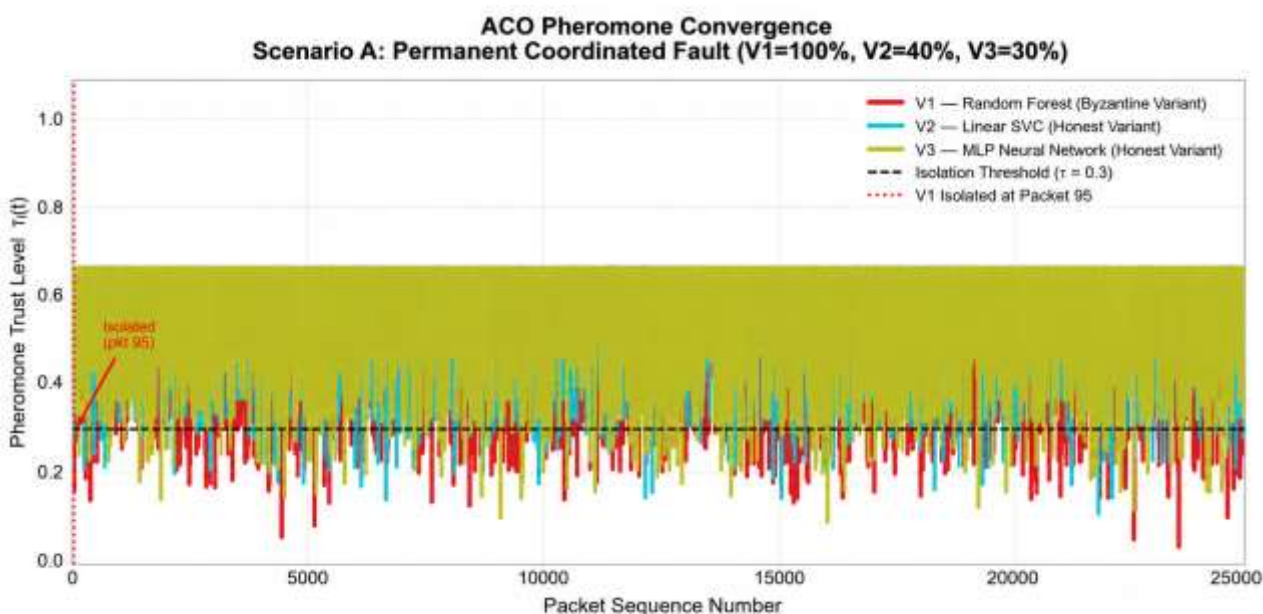


Fig. 3: Scenario A pheromone convergence for ACO-Voter. The Byzantine variant V1 (red) is isolated below the threshold $\theta = 0.30$ at packet 95, while the honest variants V2 and V3 converge toward the equilibrium $\tau^* \approx 0.667$.

Fig. 3 shows the pheromone trajectories of the three variants: the Byzantine variant V1 is isolated below the threshold $\theta = 0.30$ within 95 packets of fault onset (0.38% of the 25,000-packet test stream), while the honest variants V2 and V3 converge toward the equilibrium value $\tau^* \approx 0.667$ given by Eq. (6). At a representative IoT gateway throughput of 10,000 packets/second, isolation within 95 packets corresponds to roughly 9.5 milliseconds of fault onset. Once isolated, V1's diminished pheromone weight prevents it from pulling the weighted consensus toward incorrect normal classifications, whereas SMV - which assigns equal trust throughout - cannot adapt and therefore misses these additional attacks. No static voter possesses an equivalent isolation capability, since SMV and FWV's weights never change regardless of variant behaviour.

Under this scenario, ACO-Voter matches FWV's recall exactly (0.634) and slightly exceeds its F1-score (0.768 vs. 0.758), despite FWV's weights being pre-configured with oracle knowledge that V1 is the most vulnerable variant - an assumption unavailable in real deployments. If V2 or V3 were instead the degraded variant, FWV's fixed weighting would offer no benefit and could even prove counter-productive, whereas ACO-Voter adapts identically regardless of which variant degrades, since it identifies faulty behaviour empirically rather than assuming it in advance. FWV's marginally higher accuracy (0.901 vs. 0.854) chiefly reflects the true-negative-heavy composition of the dataset (72.6% normal traffic); on recall - the primary metric for security-critical detection - the two are identical. EMA-Voter, whose multiplicative update rule ties recovery and isolation speed to the current weight level, adapts too slowly within the test-stream duration to differentiate itself from SMV, producing metrics identical to the SMV baseline (Recall 0.464, F1 0.621).

5. CONCLUSION AND FUTURE WORK

This paper proposed ACO-Voter, a pheromone-based adaptive trust-weighting mechanism for NVP-based Intrusion Detection Systems at the IoT edge layer. Three independently implemented classifier variants - Random Forest, Linear SVC, and MLP - are each assigned a pheromone trust value that is reinforced on agreement and evaporated on disagreement with the ensemble consensus, enabling autonomous fault isolation without prior knowledge of which variant will fail. Under a coordinated permanent Byzantine fault scenario on the DNN-EdgeIoT dataset, ACO-Voter isolated the faulty variant within 95 packets and improved attack recall from 0.464 to 0.634, a 36.6% relative gain over SMV, while matching the performance of an oracle-informed fixed-weight baseline without requiring any prior knowledge of which variant would fail.

This paper forms the basis of an extended study, currently in preparation, which evaluates ACO-Voter under intermittent and recovery-type Byzantine fault conditions, formally validates the improvement using 10-fold cross-validation and non-parametric significance testing, and reports a sensitivity analysis of the pheromone evaporation rate ρ . Future work will also extend the architecture to $N > 3$ variants and incorporate contextual trust factors such as node mobility in dynamic IoT topologies.

ACKNOWLEDGEMENT

The authors declare that no funding or grants of any kind were received in support of this research, and that they have no competing interests to declare.

REFERENCES

- [1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347-2376, 2015.
- [2] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "EdgeIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications," *IEEE Access*, 2022.
- [3] M. L. Shooman, "Validating software reliability models," *IFAC Proceedings Volumes*, vol. 21, no. 15, pp. 115-121, 1988.
- [4] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable and Secure Computing*, vol. 1, pp. 11-33, 2004.
- [5] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proc. IEEE IJCNN*, vol. 2, pp. 1702-1707, 2002.
- [6] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11994-12000, 2009.
- [7] J. Gu and S. Lu, "An effective intrusion detection approach using SVM with Naïve Bayes feature embedding," *Computers & Security*, vol. 103, p. 102158, 2020.
- [8] R. Reischuk, "A new solution for the Byzantine generals problem," *Information and Control*, vol. 64, no. 1, pp. 23-42, 1985.
- [9] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI '99*, New Orleans, LA, pp. 173-186, 1999.
- [10] M. M. Aung and Y. S. Zaw, "N-version intrusion detection system with decision fusion," in *Proc. ICICT*, pp. 1-6, 2019.
- [11] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, 2nd ed. Wiley, 2014.
- [12] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and Computation*, vol. 108, no. 2, pp. 212-261, 1994.
- [13] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evolutionary Computation*, vol. 1, no. 1, pp. 53-66, 1997.
- [14] A. Abraham, R. Jain, J. Thomas, and S. Y. Han, "D-SCIDS: Distributed soft computing intrusion detection system," *Journal of Network and Computer Applications*, vol. 30, no. 1, pp. 81-98, 2007.