Enhancing IoT object integrity and energy efficiency through DBSCAN and fuzzy Logic-based self-management framework

Abdelhamid Garah, Nader Mbarek, Sergey Kirgizov

Journal of Systems Architecture 175 (2026) 103751

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Enhancing IoT object integrity and energy efficiency through DBSCAN and fuzzy Logic-based self-management framework☆

Abdelhamid Garah *, Nader Mbarek, Sergey Kirgizov

LIB UR 7534, Université Bourgogne Europe, Dijon, F-21000, France

ARTICLE INFO

Keywords:
Internet of Things (ioT)
IoT object integrity
Lightweight remote attestation
Autonomic Computing
Fuzzy Logic
DBSCAN

ABSTRACT

Ensuring the integrity of Internet of Things (IoT) objects is challenging due to their limited energy and processing resources, as well as their exposure to security threats. Remote Attestation (RA) is a widely used technique that enables a trusted entity, such as a gateway, to verify the integrity of constrained IoT devices remotely. However, applying RA in constrained environments introduces challenges, including redundant attestations, high energy consumption, and vulnerabilities, such as Time-of-Check-Time-of-Use (TOCTOU) attacks. To address these limitations, this paper proposes a novel autonomic IoT framework for self-managing the integrity of IoT objects using a lightweight remote attestation mechanism and the Autonomic Computing paradigm. The proposed approach uses a DBSCAN model to determine when attestation is required, along with a fuzzy-logic system that dynamically selects an appropriate lightweight hash function based on the device state. Meanwhile, the attestation process uses a lightweight HMAC scheme to ensure device integrity. Our proposed framework reduces redundant attestations, optimizes energy consumption, and extends the lifetime of IoT systems, making it suitable for resource-constrained environments.

1. Introduction

The Internet of Things (IoT) is growing rapidly, leading to the deployment of large-scale devices across domains such as smart homes, industry, e-health, and smart cities. These devices continuously collect and exchange valuable information, making security a critical requirement for the proper and reliable operation of such systems. Among these security requirements, ensuring the integrity of IoT devices is particularly important because compromised devices can cause operational failures, data manipulation, service disruptions, and other issues. Ensuring the integrity of IoT devices presents different challenges due to the extensive use of low-power, limited-computing, and heterogeneous technology devices. Therefore, Remote attestation (RA) is a well-known method for ensuring the integrity of resource-constrained devices by enabling remote integrity verification. With RA, an IoT device (prover) sends its system measurements to a remote verifier, such as a gateway or a cloud-based entity, that confirms the device's integrity based on stored information [1].

1.1. Motivation

To ensure the integrity of IoT devices, several RA schemes have been proposed [2–11]. Although existing RA approaches provide integrity verification, most of these techniques rely on a periodic attestation method that does not consider the state of the devices or their environment. Such methods cause significant energy waste and shorten the lifetime of IoT devices due to redundant attestation processes across devices. Indeed, each attestation round requires that all devices undergo a verification process regardless of whether attestation is needed or not. Such redundancy imposes a uniform attestation period across all devices, thereby increasing energy consumption.

Furthermore, security measures are chosen without sufficient regard for these IoT objects, which can affect performance and scalability. In addition, several challenges remain to be overcome, including Time-Of-Check-Time-Of-Use (TOCTOU) attacks, Verifier-impersonation, Denial of Service (DoS) attacks, and transient & self-relocating malware [1].

These limitations highlight the need to develop adaptive and lightweight remote attestation mechanisms that can effectively perform integrity checks without incurring significant resource costs.

1.2. Contributions

To address the above challenges, we propose a novel autonomic IoT framework that uses the Autonomic Computing paradigm and a

☆ This article is part of a Special issue entitled: 'AI4ORC' published in Journal of Systems Architecture.

* Corresponding author.
E-mail addresses: Abdelhamid.Garah@u-bourgogne.fr (A. Garah), Nader.Mbarek@u-bourgogne.fr (N. Mbarek), Sergey.Kirgizov@u-bourgogne.fr (S. Kirgizov).

https://doi.org/10.1016/j.sysarc.2026.103751
Received 14 January 2026; Received in revised form 14 February 2026; Accepted 18 February 2026
Available online 27 February 2026

lightweight Remote Attestation process to ensure the integrity of IoT objects. The main contributions of this work are summarized as follows:

- We propose an autonomic IoT integrity management framework that enables self-managed and energy-aware remote attestation.
- Our approach consists of two phases: the IoT selection phase and the attestation verification phase.
- In the selection phase, we employ a Density-Based Spatial Clustering of Applications with Noise (DBSCAN) model to determine when an IoT object requires attestation, reducing redundant attestation operations.
- We also design a fuzzy logic system in the selection phase to dynamically select suitable lightweight hash functions based on the device state.
- In the verification phase, we implement a lightweight HMAC-based attestation process to ensure device integrity while optimizing energy consumption and extending the lifetime of IoT systems.

The rest of this paper is structured as follows: Section 2 reviews related work. Section 3 presents our self-management framework. Section 4 describes the DBSCAN model used to select IoT objects requiring attestation verification, while Section 5 explains the Fuzzy Logic system for dynamically choosing the most suitable lightweight hash functions. Section 6 discusses the evaluation scenarios and results. Finally, Section 8 concludes the paper and outlines potential future research directions.

2. Related works

2.1. Remote attestation for IoT objects integrity

Remote Attestation (RA) allows the IoT device (the prover) to prove its Integrity to a remote entity (the verifier), such as a gateway or a cloud-based entity. A variety of IoT Remote Attestation solutions have been proposed, which are usually classified into three categories: software, hardware, and hybrid RA schemes [1,12].

For example, for software-based Remote Attestation schemes, researchers employ software techniques to ensure the IoT objects' integrity, including memory filling [3], Pseudo-random Memory Traversal (PMT) [2], strict response time mechanisms [4], and attestation based on random construction of the attestation function [2]. On the other hand, there are some hardware-based remote attestation schemes. Researchers have used dedicated hardware modules including Physical Unclonable Functions (PUFs) [6], Trusted Platform Modules (TPMs) [5], and Trusted Execution Environments (TEEs) [7,8]. These hardware-based solutions provide robust security, but are more expensive to deploy. To address the limitations of both approaches, hybrid attestation schemes [9–11] have been developed that combine the strengths of software-based and hardware-based techniques.

In this research, we propose a lightweight software-based Remote Attestation approach that utilizes Fuzzy Logic and the Autonomic Computing paradigm to ensure the integrity of IoT objects.

2.2. Autonomic Computing and IoT security

The Autonomic Computing (AC) paradigm enables systems to self-manage and adapt themselves with minimal or no human intervention. It is based on two fundamental elements: the Autonomic Manager and the Managed Resources, as shown in Fig. 1. For IoT systems, the AC components are projected as follows: The IoT objects represent the Managed Resources component, equipped with two interfaces: Sensors to gather data and Effectors to execute actions. However, the Autonomic Manager component implements the MAPE-K closed control loop (Monitor, Analyze, Plan, Execute, and Knowledge plane) to manage the these IoT objects [13].

Limited research has proposed the use of the Autonomic Computing paradigm to ensure IoT security services. Almeida et al. [14] propose a solution that combines AC with dendritic cell algorithms to detect potential IoT threats efficiently. Another solution [15] aims to strengthen IoT security with little human involvement. In this solution, an ontology-based autonomic manager continuously monitors, analyzes, and responds to security attacks by planning and executing adaptive security actions. Abdelhamid et al. [16] proposed a solution that combines the AC paradigm with lightweight cryptography to ensure the confidentiality of IoT data.

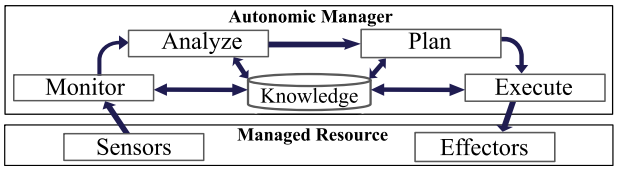

Fig. 1. The Autonomic Computing components.

3. Framework for IoT objects integrity self-management

In the following sections, we present our Autonomic IoT security architecture, a comprehensive overview of the interactions between the key components that enable self-management of IoT devices' integrity in our proposed Autonomic IoT security architecture, and a detailed description of the MAPE-K closed control loops that we implement to ensure the ongoing integrity of IoT devices without human intervention.

3.1. Proposed autonomic IoT architecture

We have designed an innovative Autonomic Computing-based IoT security architecture that consists of three layers: Cloud, Network, and Sensing layers (see Fig. 2). The Cloud layer provides services such as big data analysis, storage, and visualization, among others. Below this layer, the Network layer, which serves as a link between the Cloud and Sensing layers, enables the transmission of data between IoT devices and the cloud. Finally, the Sensing layer is divided into two sub-layers, as shown in Fig. 2. The first sub-layer, known as the IoT Devices, is responsible for collecting data from the IoT environment, forwarding it to IoT gateways and the cloud, and executing user commands via the IoT application. The second sub-layer is the Autonomic Edge Security. This includes two different types of IoT security gateways. The Low-Level Security Gateway (LL-SG) enables data communication between IoT objects and the second security gateway. The LL-SG also enables security measures, such as authentication and controls, and manages similar IoT devices within the same cluster. Furthermore, the High-Level Autonomic Security Gateway (HL-ASG) implements the MAPE-K control closed-loop to manage clusters of IoT objects.

3.2. IoT sensing layer components interactions

In the following subsection, we present a Message Sequence Chart (MSC) (see Fig. 3) to provide a graphical representation of the interactions within the sensing layer, highlighting how these components allow the selection of suspicious devices using a DBSCAN model and then attest them in the attestation phase through a dynamically changing lightweight attestation approach that adjusts the lightweight hash function based on a Fuzzy Logic system.

As illustrated in Fig. 3, once the IoT Object is authenticated and shares its secret key with the HL-ASG, it receives a state request from the LL-SG. The IoT Object encrypts its current state (i.e., residual energy, application type, utilization ratio) using the shared key (Ks). This encrypted state is then sent to the HL-ASG via the LL-SG. In the initial integrity verification phase, all IoT Objects are selected for integrity

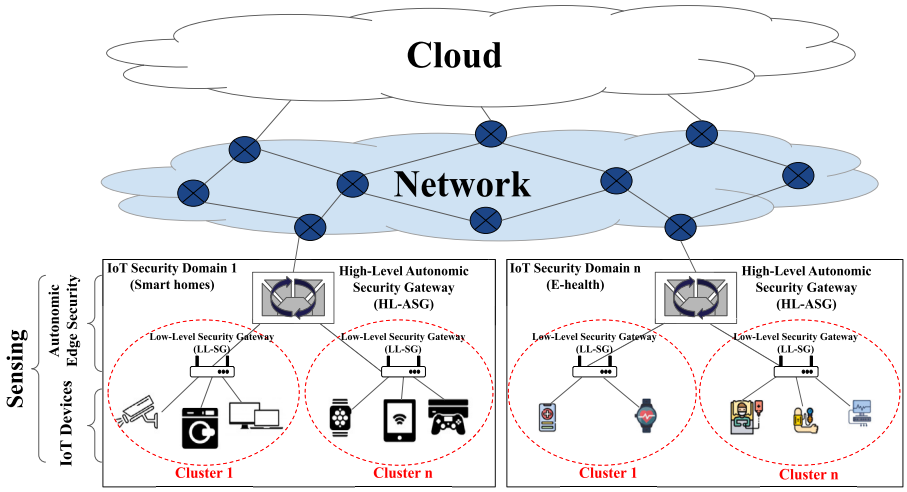

Fig. 2. The proposed autonomic IoT security architecture.

verification by default. However, in subsequent iterations, the HL-ASG utilizes the MAPE-K_1, based on the DBSCAN model, to determine whether these devices require attestation (see Section 4). Once the IoT Object is selected for attestation, the HL-ASG issues a command to the IoT Object to pause the transmission of IoT data to the cloud via the two security gateways until the attestation phase is completed. Then, the HL-ASG decrypts the received state of the IoT Object using Ks, and applies a Fuzzy Logic process via MAPE-K_1 to determine the appropriate lightweight hash function for attestation based on the state of the IoT Object (i.e., residual energy, application type, utilization ratio) (as described in Section 5). The above process corresponds to the first phase, the selection phase of our proposed lightweight autonomic remote attestation scheme, where only suspicious IoT Objects are selected for attestation along with the corresponding lightweight hash function.

After that, the HL-ASG (the verifier) generates a challenge (nonce) and stores it along with the selected lightweight hash function (H) in the knowledge plane. The challenge is then encrypted and sent to the IoT Object. Upon receiving and decrypting this attestation challenge, the IoT Object uses the challenge information: the lightweight hash function (H) and the nonce, along with the IoT object's memory contents ($M_{IoTObject}$)and the shared key (Ks), to compute a lightweight HMAC using the formula (1) [17], where ipad and opad are the inner and outer padding constants (bytes $0 \times 5C$ and 0×36 repeated B times, with B representing the block size of the lightweight hash function). This lightweight HMAC is considered as the attestation result of the IoT Object that is sent through the LL-SG to the HL-ASG for verification.

$$\text{lightweight HMAC} = H\left(\left(K_s \oplus \text{opad}\right) \| H\left(\left(K_s \oplus \text{ipad}\right) \| \left(\text{nonce} \| M_{IoTObject}\right)\right)\right) \quad (1)$$

At this stage, the attestation phase (i.e., attestation verification) begins. The second closed control loop, MAPE-K_2, manages it. In this context, the HL-ASG retrieves the necessary information from the knowledge plane to compute the HL-ASG attestation (i.e., a lightweight HMAC) for this IoT Object. Specifically, it retrieves the Ks, the nonce, H, and the stored memory content of the IoT object ($HL\text{-}ASG_{M_{IoTObject}}$). The calculated HMAC is then compared to the HMAC received from the IoT Object to validate the integrity of its software. If the attestation fails, the HL-ASG prompts the LL-SG to update the authentication with the IoT Object by denying its access. On the contrary, if the attestation is valid, the HL-ASG sends a command to resume the IoT data transmission to the Cloud via the two security gateways. The IoT Object continues this transmission until the next Selection Time is reached. This Selection Time parameter is a hardware-enforced parameter defined by a secure, read-only clock that prevents tampering or attacks [18].

When the next Selection Time is reached, the IoT Object sends its encrypted state (i.e., packet rate, energy consumption, utilization ratio, memory usage, residual energy, and application type) to the HL-ASG via LL-SG. The first phase, the selection phase, starts again. The HL-ASG decrypts this IoT Object state and uses a portion of it (i.e., packet rate, energy consumption, memory usage, and utilization ratio) as input to the DBSCAN model of MAPE-K_1 (described in Section 4) to determine whether the IoT Object should be attested. If the IoT Object is not selected for attestation, its attestation remains valid. The IoT Object continues to encrypt and transmit the IoT data to the cloud until the next Selection Time is reached. The Ks used for encryption is frequently updated to protect the confidentiality of the IoT data. Conversely, if the IoT Object is selected for the attestation, the HL-ASG orders the IoT Object to pause data transmission until the attestation phase is achieved. Next, the HL-ASG applies a Fuzzy Logic model via MAPE-K_1 (as described in Section 5) to select the appropriate lightweight hash function for attestation based on the last portion of the IoT Object's state (i.e., residual energy, application type, and utilization ratio). Algorithm 1 provides a detailed description of the selection phase.

After that, the attestation phase begins. The HL-ASG defines a challenge by generating a nonce, storing it with the lightweight hash function in the knowledge plane, and encrypting this challenge before sending it to the IoT Object via the LL-SG. The IoT Object decrypts the challenge, and computes the attestation using a lightweight HMAC. This attestation result is sent through the LL-SG to the HL-ASG for verification. The attestation verification is managed by MAPE-K_2 as shown in Algorithm 2.

If the attestation fails, the HL-ASG updates the authentication by denying connection with the IoT Object via a prompt to the LL-SG. The IoT Object can perform an additional authentication process and reinitiate the integrity verification, thereby resolving the problem. On the contrary, if the attestation is valid, the IoT Object resumes IoT data transmission after receiving a prompt from the HL-ASG, allowing it to encrypt and transfer the IoT data to the Cloud. This IoT data transmission continues until the next Selection Time, when another integrity verification is triggered. This iterative process forms a closed control loop between the IoT Object, LL-SG, and HL-ASG, enabling autonomic remote attestation through a two-step selection and attestation phases.

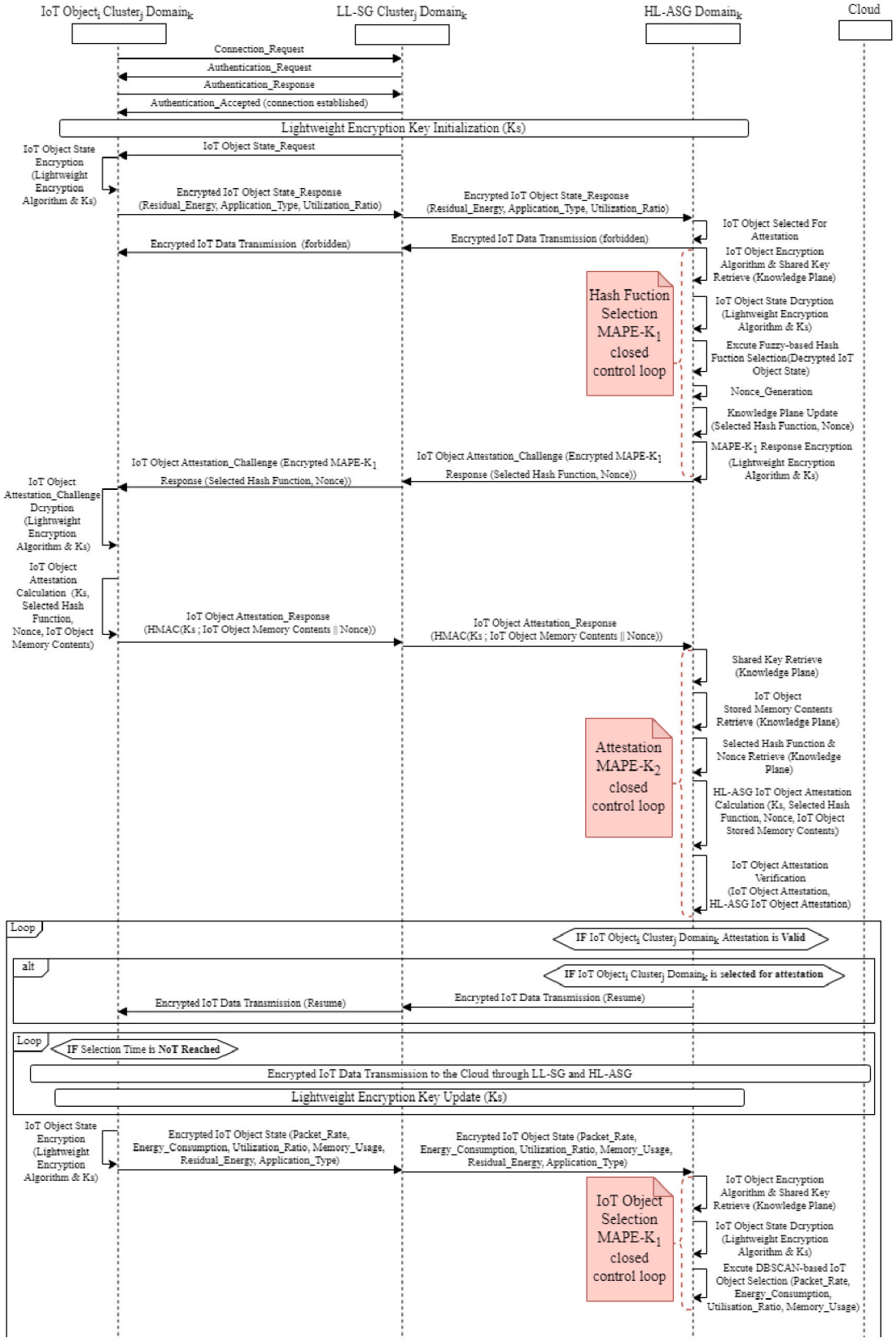

Fig. 3. MSC for IoT sensing layer integrity self-management.

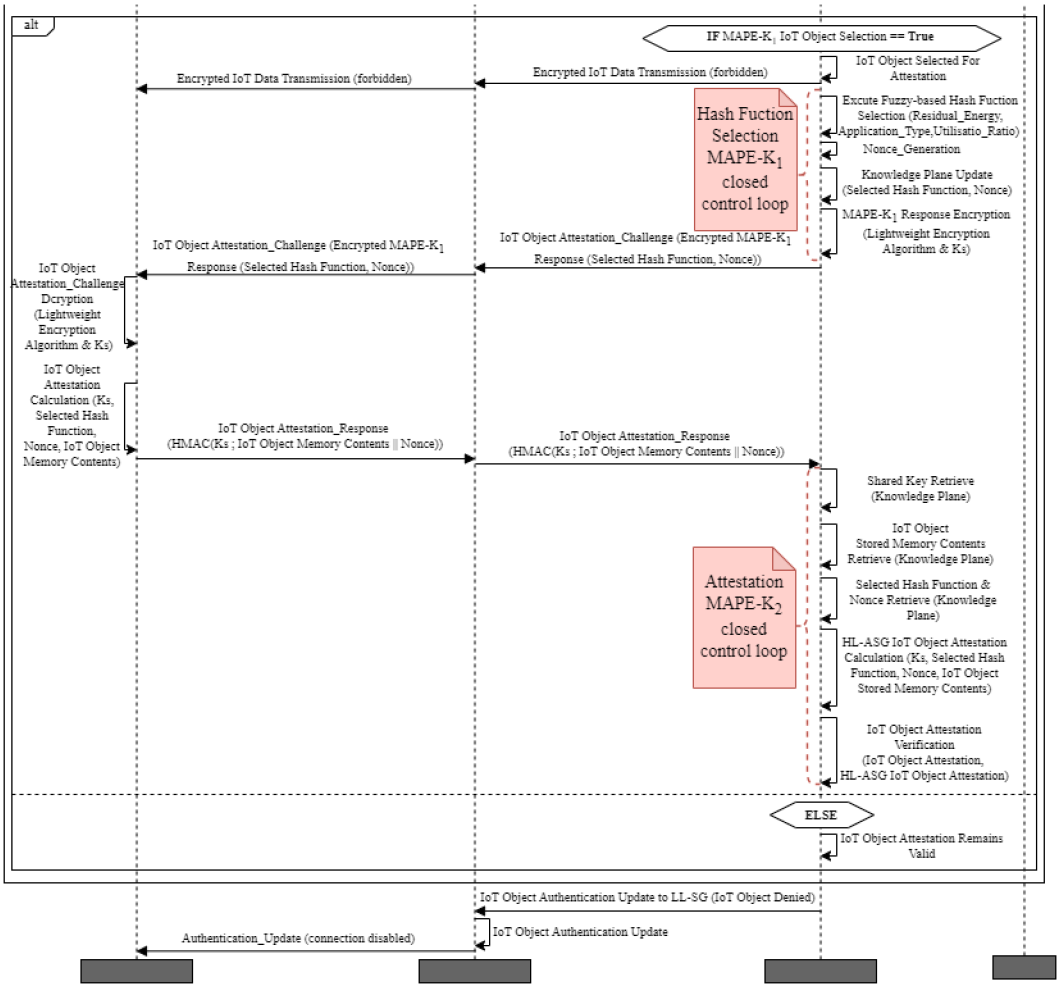

Fig. 3. (*continued*).

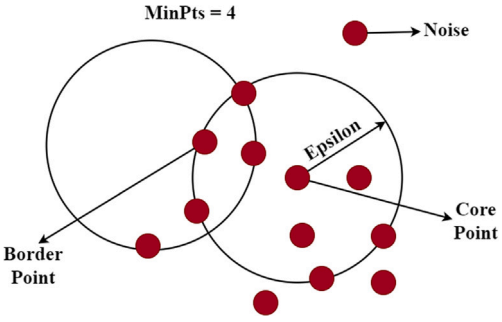

Fig. 4. DBSCAN point types.

4. DBSCAN-based IoT object selection for attestation process

During the Analyze phase of the MAPE-K_1 selection closed control loop that we implement within the HL-ASG of our proposed architecture, we opted to use the DBSCAN algorithm to determine if an IoT object requires attestation. DBSCAN is a robust clustering method that does not require a predetermined number of clusters. This enables the clustering of IoT devices with similar characteristics and the identification of outliers that require attestation.

Because a suitable dataset was not available in the literature, we collected the necessary data to train our DBSCAN model through simulations, as detailed below.

4.1. Dataset collection

The primary objective of our dataset creation is to collect simulation data from various operational scenarios and attack conditions, utilizing the OMNeT++ platform and the INET framework [19,20]. However, we have implemented two new models due to some limitations of the INET framework regarding IoT security and self-management. The first module implements lightweight block ciphers using the Crypto++ library [21]. The second module focuses on the energy consumption and memory usage (RAM) of these ciphers, which are based on an earlier study [22]. This module provides insight into the power consumption and RAM usage of the IoT devices used in our simulation.

Thus, our simulated architecture consists of three clusters connected to an HL-ASG. Each cluster has a single LL-SG wirelessly connected to a group of IoT devices using the IEEE 802.11ac standard. Moreover, we use a wired connection between the HL-ASG and the LL-SGs. Each IoT device belonging to the same cluster runs the same type of IoT applications, which are classified into Non-Critical (NC), Critical (C), and Very Critical (VC) applications, based on their security requirements and role in critical operations, ranging from the lowest to the highest security levels needed. Additionally, we utilize the same packet traffic payloads (1024-byte payloads) across all IoT devices. Furthermore, all IoT devices are of the same type: a Raspberry Pi 3 powered by a 3.7 V, 200 mAh battery providing 2664 joules (J) of power.

During an attack, an IoT device may exhibit anomalous behavior, manifesting as significant deviations from typical performance metrics, such as high energy consumption, packet rate, memory usage, and

Algorithm 1: The MAPE-K_1 selection phase

Input:

attestationSelectionVector, /* [packetRate, energyConsumption, utilizationRatio, memoryUsage] */
hashFunctionSelectionVector /* [residualEnergy, applicationType, utilizationRatio]*/

Output:

attestationDecision, // {True, False}
selectedHashFunction, // lightweight hash function
nonce, // random nonce
attestationChallenge // [selectedHashFunction, nonce]

```
attestationDecision ← DBSCAN(attestationSelectionVector);
if (attestationDecision == False) then
    // IoT Object is not selected for attestation
    Return attestationDecision;
else
    // IoT Object is selected for attestation
    selectedHashFunction ← FuzzyLogic(hashFunctionSelectionVector);
    nonce ← generateNonce();
    attestationChallenge ← [selectedHashFunction, nonce];
    Return attestationDecision, attestationChallenge;
end
```

Algorithm 2: MAPE-K_2 attestation verification.

Input:

hmacIotObject,
ks, // The shared key
nonce, // a random nonce
selectedHashFunction, // a lightweight hash function
iotObjectmemoryContentHlasg /* The memory contents of the IoT object stored in the Knowledge plane of the HL-ASG.*/

Output: attestationResult // {True, False}

```
hmacHlasg ← hmacCalculation(ks, nonce, selectedHashFunction, iotObjectmemoryContentHlasg);
if (hmacIotObject == hmacHlasg) then
    // IoT object integrity is valid
    attestationResult = True;
else
    // IoT object integrity is compromised
    attestationResult = False;
end
Return attestationResult;
```

utilization ratio. The construction of the DBSCAN model was guided by the following four parameters (i.e., features), as determined by this discovery.

Table 1
Description of the dataset obtained from the simulation.

	Packet rate (packets/s)	Energy consumption (%)	Utilization ratio (%)	Memory usage (%)
Mean	1.50	1.80	68.89	4.92
Std	3.49	1.23	44.39	3.18
Min	0	0.49	0	0
Max	33.34	9.05	100	8.4

- **Packet Rate:** it defines the number of packets transmitted per second.
- **Energy Consumption:** it is the system's energy consumption percentage in a time interval.
- **Utilization Ratio:** it represents the percentage of time the IoT object is active and not idle in a time interval.
- **Memory Usage:** this parameter defines the percentage of the memory the IoT object uses in a time interval.

Three modes were considered while simulating different operational scenarios for the lifetime of IoT devices: always ON, fixed ON/OFF, and dynamic ON/OFF. These modes reflect how an IoT device alternates between active and idle states of usage. Additionally, we simulated attack situations to create abnormal (i.e., outlier) scenarios, including Denial of Service (DoS) attacks, Data Injection attacks, Physical Tampering, etc.

With only four parameters (i.e., features), the simulations produced 4458 samples, which is a medium dataset. A statistical overview of our dataset is presented in Table 1, which includes the mean, standard deviation (Std), lowest, and highest observed values.

Finally, we performed data normalization (scaling) thanks to the StandardScaler from the scikit-learn Python library [23]. The normalization is based on the formula (2):

$$\text{scaled value} = \frac{\text{original value} - \text{mean of the feature}}{\text{standard deviation}}. \quad (2)$$

Normalization is an essential preprocessing step for many machine learning algorithms, including DBSCAN, which are based on some distance metrics (e.g., Euclidean). If the features are on different scales, for example, the value of packet rate and the percentage of energy consumption, then features with larger scales (i.e., packet rate) will dominate in the learning process. Indeed, normalization ensures that all the features contribute equally to the model by usually transforming them to have a mean of 0 and a variance of 1. This way, it is easier for the model to perform meaningful distance calculations, and the model converges faster.

4.2. DBSCAN model training and inference process

The DBSCAN algorithm is designed to identify clusters based on the density of samples (i.e., data points), thereby ensuring efficiency in detecting arbitrarily shaped clusters and automatically detecting noise. This approach is well suited for datasets with clusters of varying shapes and sizes, and it does not necessarily require that the number of clusters be predefined in advance. To build a DBSCAN model, two most important parameters must be defined:

- **Epsilon** (ϵ): the maximum distance between two data points for them to be considered neighbors.
- **MinPts:** the minimum number of neighbors (data points) in the neighborhood for a data point to be considered as a core point [24].

By setting these parameters, the DBSCAN algorithm classifies points into three main types (see Fig. 4):

- **Core Point:** a point p is a core point if it has at least MinPts points, including itself, within a distance ϵ.

- **Border Point:** a point p is a border point if it has less than MinPts points within distance ϵ, but it is in the neighborhood of a core point q. Thus, it is not a core point but within ϵ distance of one.
- **Noise (Outlier):** a point p is considered noise if it does not belong as either a core point or a border point (it is not within ϵ of any core point) [24].

Before specifying the steps of the DBSCAN clustering algorithm, it is essential to define some additional concepts that the algorithm will use to form clusters:

- **Directly Density-Reachable:** a point q is directly density-reachable from a core point p if q is within the distance ϵ from p, and p is a core point.
- **Density-Reachable:** a point q is density-reachable from point p if there exists a chain of points $p_1, p_2, \ldots, p_n$, where $p_1 = p$ and $p_n = q$, and each point in the chain is directly density-reachable from the previous one.
- **Density-Connected:** two points p and q are density-connected if there exists a point o such that both p and q are density-reachable from o [24].

The DBSCAN algorithm processes all points in the training dataset, assigning each point a label upon its visit. A point is designated as a core point if it has a minimum of MinPts neighbors within a radius of ϵ. In the case that a point is determined to be a core point, the initial class is initiated (class 1). The core point and its neighbors within distance ϵ are labeled as belonging to class 1. Subsequently, the algorithm performs a recursive search to identify all density-connected points and assigns them to class 1. This process is fundamental to the evolution of the class. It is important to note that, once all density-reachable points have been visited, class 1 stops growing. The DBSCAN algorithm will proceed to revisit unvisited points, and a new class will be initiated as soon as it encounters another core point. This class can be expanded similarly. Finally, all points not allocated to a specific class are considered noise points.

After training our model thanks to our generated dataset and the scikit-learn Python library [23], we obtained five clusters different from the label −1, as well as a set of points labeled as −1 (i.e., outliers) signifying that they do not belong to any cluster. In our study, we have combined the five clusters into a single cluster, which represents the group of IoT devices that were not selected for attestation verification. The remaining points, designated as "−1" labels, correspond to the outliers. These points represent the IoT devices that have been selected for attestation verification. The clustering results are then represented in a two-dimensional plot, as shown in Fig. 5. Given that the dataset under consideration contains four input parameters (i.e., packet rate, energy consumption, memory usage, and utilization ratio), corresponding to four dimensions, t-SNE (t-Distributed Stochastic Neighbor Embedding) was applied for dimensionality reduction. t-SNE is an unsupervised, non-linear dimensionality reduction technique that is commonly used for exploring and visualizing high-dimensional data [25]. However, as a result of the dimensional reduction, points corresponding to IoT devices that were not selected for attestation and the outlier points (i.e., IoT devices selected for attestation) appear to be nearby, but they are not.

Following the construction of the DBSCAN model, the next step involves the classification of a new data point, denoted by (x). Initially, it is necessary to perform data normalization (i.e., scaling) on the new sample x. Then, it is essential to identify all neighbors within ϵ-distance of x. If this new point (i.e., x) is a neighbor of one of the core points from the dataset, it will be assigned to the same class as the core point. In the absence of a core point within the neighborhood of x, it is designated an outlier point.

4.3. DBSCAN model evaluation

Evaluating the performance of DBSCAN or any clustering algorithm, involves assessing the quality and validity of the clusters it forms. Below are some standard metrics that are commonly used to validate these types of algorithms and that we use to evaluate our proposed DBSCAN model enabling IoT object selection for autonomic attestation [26,27]:

1. **Silhouette Coefficient:** it is a metric that describes the similarity of an object to its own cluster compared to other clusters. This value ranges from −1 to +1. A higher value indicates good clustering. For each point i the Silhouette Coefficient $S(i)$ is calculated using Formula (3):
$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{3}$$
where for each point i, $a(i)$ is the average distance between i and all other points in the same cluster, and $b(i)$ is the average distance between i and points in the nearest cluster to which i does not belong.
2. **Davies–Bouldin Index (DBI):** it gives the average similarity ratio of every cluster to its most similar cluster. The lower values signify better clustering. It is calculated using Formula (4):
$$DBI = \frac{1}{n} \sum_{i=1}^{n} \max_{i \neq j} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) \tag{4}$$
where σ_i is the average distance between each point in cluster i and its centroid, $d(c_i, c_j)$ denotes the distance between the centroids of clusters i and j, and n is the total number of clusters.
3. **Adjusted Rand Index (ARI):** it performs the task of calculating the similarity between two clustering results by comparing the number of pairs of points that are assigned to the same or different clusters in each clustering. It varies from −1 to 1, where perfect matches have a value of 1. It is calculated using Formula (5):
$$ARI = \frac{RI - \text{Expected RI}}{\max(\text{RI}) - \text{Expected RI}} \tag{5}$$
where RI (Rand Index) is the number of point pairs that are either in the same cluster or in different clusters in both true and predicted clustering. Expected RI is the expected value of RI under the assumption of random clustering.
4. **Cluster Purity:** it is the simple measure of how well the clusters contain only one class of data points. In other words, it measures how well the clustering model categorizes samples of the same true class into the same cluster. Cluster Purity ranges from 0 to 1, where 1 means that each cluster contains samples from a single class. It is calculated using Formula (6):
$$\text{Purity} = \frac{1}{N} \sum_{i=1}^{k} \max_{j} |C_i \cap L_j| \tag{6}$$
where N represents the total number of samples, C_i denotes the set of samples in cluster i, L_j refers to the set of samples in true class j, and $|C_i \cap L_j|$ indicates the number of samples from class j in cluster i.

According to the definitions provided, the results of the DBSCAN model evaluation are presented in Table 2. We observe that our model attains a Silhouette Coefficient value of 0.72, indicating good separation between clusters in our model. Generally, values above 0.5 are considered reasonable. Furthermore, the Davies–Bouldin index, a measure of cluster separation, was determined to be 0.73, indicating that the clusters were well separated. This finding aligns with the results of the Silhouette Coefficient analysis. Finally, the values obtained for the Adjusted Rand Index and Cluster Purity are 1, representing perfect agreement between the predicted clusters and the true class labels (i.e., selection of IoT objects or not for attestation verification) as well

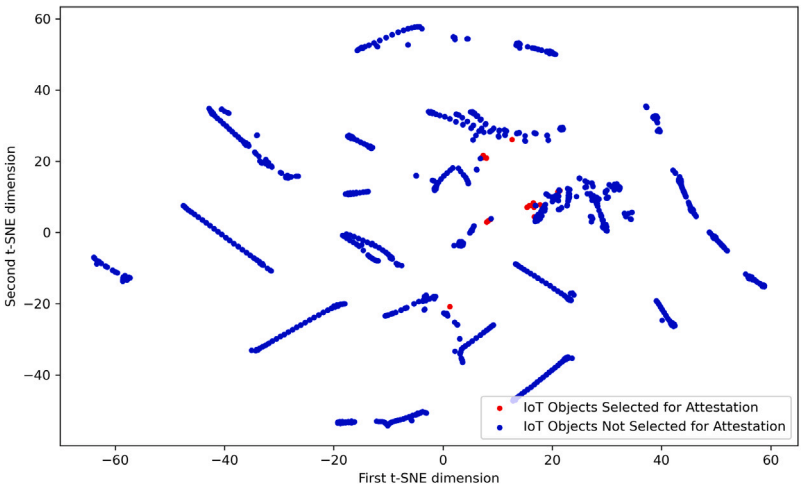

Fig. 5. 2D visualization of our DBSCAN clustering results.

Table 2
DBSCAN evaluation results.

Metrics	Values
Silhouette Coefficient	0.72
Davies–Bouldin Index	0.73
Adjusted Rand Index	1.0
Cluster Purity	1.0

as each cluster has data points from only one class, which means perfect clustering with respect to class labels.

In general, these are promising results for our DBSCAN model. A high value for the Silhouette Coefficient and a low value for the Davies–Bouldin Index, along with perfect values for the ARI and Cluster Purity metrics, indicate that the model does an excellent job of identifying IoT objects that require attestation verification to ensure the integrity of those IoT objects.

5. Fuzzy-based IoT objects integrity verification

In order to select a lightweight hash function for the attestation verification phase based on IoT residual energy, application type, and usage ratio, we integrated Fuzzy Logic, specifically the Mamdani Fuzzy Inference System (FIS), with the MAPE-K closed control loop. Fuzzy Logic is a simple and interpretable rule-based method that is suitable for IoT contexts where energy limitations are present, data may be noisy or incomplete, and response latency is critical. Furthermore, Fuzzy Logic models complex nonlinear relationships and emulates human reasoning [28].

The Mamdani FIS is comprised of four steps: fuzzification, rule evaluation, aggregation, and defuzzification. In the following section, an overview of each phase is presented in the context of the proposed IoT object integrity self-management system (see Fig. 6).

5.1. Fuzzification

Fuzzification is the process of converting crisp (i.e., exact or binary values: 0 or 1) inputs into a fuzzy set membership degree, which is a value in the range of 0 and 1. The implementation of this process is made possible by a membership function. Mathematically, $\mu_A(x)$ is the membership function of a fuzzy set A, which is defined as the degree to which a crisp input x belongs to the set A. There are several membership functions, including Gaussian, singleton, and triangular [29].

In our fuzzy logic model for selecting the appropriate lightweight hash function for attestation verification, three parameters concerning IoT objects are used as inputs (Residual Energy, Application Type, and Utilization Ratio), and the output is a Lightweight Hash Function. In the following section, we provide a detailed description of these inputs, their corresponding outputs, and the membership functions associated with them.

- **Residual Energy:** This input parameter represents energy remaining in the battery of an IoT object. These values are typically expressed as a percentage, ranging from 0% (i.e., an empty battery) to 100% (i.e., a fully charged battery).
- **Application Type:** This input parameter is used to determine the criticality level of an application running on an IoT object.:
  - **Very Critical (VC):** IoT applications that require high security requirements.
  - **Critical (C):** IoT applications that also require high-security requirements but are not as strict as those for VC IoT applications.
  - **Non-Critical (NC):** IoT applications that deal with less sensitive data and have lower security requirements than the VC and C IoT applications.
- **Utilization Ratio:** This input parameter provides a quantitative representation of the frequency at which IoT objects transmit data, ranging from 0%, indicating a state of complete absence of transmission, to 100%, denoting a constant and uninterrupted transmission state.
- **Lightweight Hash Function:** This output parameter is a categorical variable representing the lightweight hash function that can be selected. Table 3 shows the hash functions used in our case, along with their characteristics [30].

In the following, we will specify the membership functions corresponding to the input and output parameters that were used in our proposed approach [29]:

1. **Residual Energy:** In the context of Residual Energy input, the Triangular membership function is a suitable option due to its capacity to provide clear and distinct transitions between low, medium, and high energy states. This function is defined by three parameters (m, n, k), and is specified by the Formula (7):

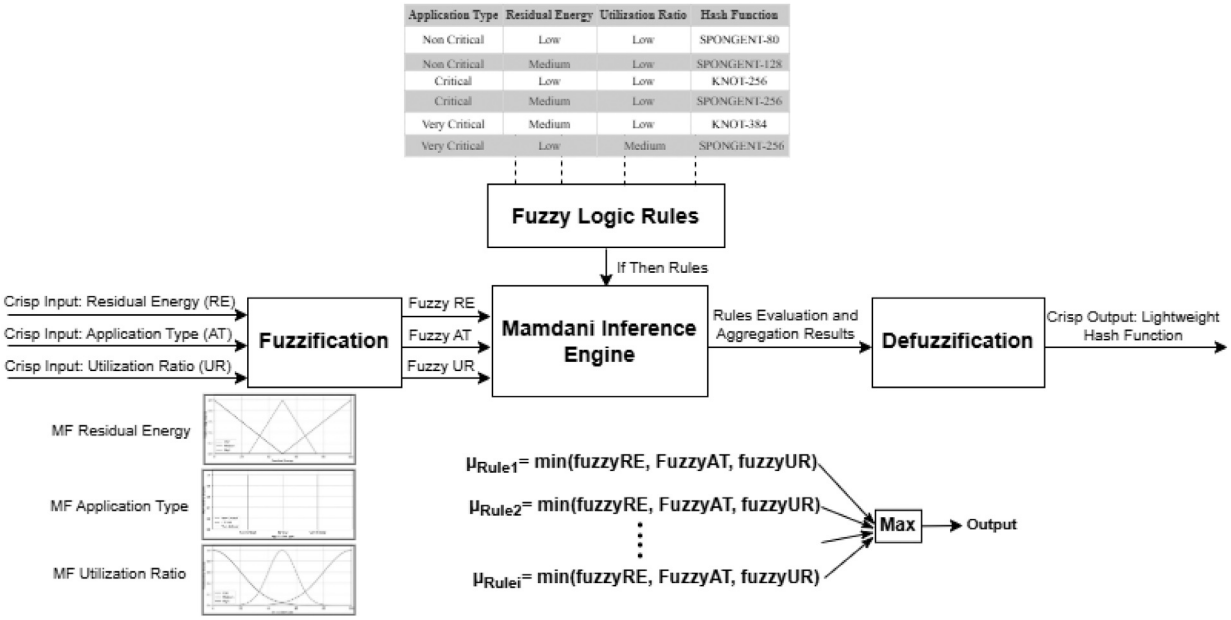

Fig. 6. Our mamdani fuzzy inference-based system steps.

Table 3
The selected lightweight hash function [30].

Algorithm	Hash length (bits)	Block size (bits)	Internal state (bits)	Throughput (kb/s)	Power (μW)
KNOT-512	512	64	512	365	8.24
KNOT-384	384	48	384	369	6.22
KNOT-256	256	128	384	1280	6.38
QUARK-256	256	32	256	3.13	4.35
SPONGENT-256	256	16	272	0.17	4.21
QUARK-176	176	16	176	2.27	3.1
QUARK-136	136	8	136	1.47	2.44
SPONGENT-128	128	8	136	0.34	2.2
SPONGENT-80	80	8	88	35.8	1.57

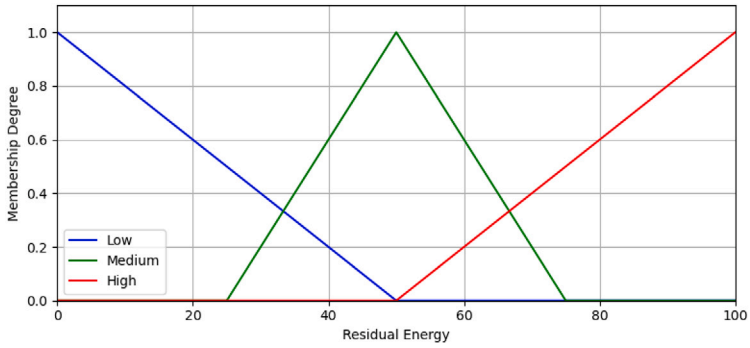

Fig. 7. The Residual Energy membership function.

$$\mu(x; m, n, k) = \begin{cases} 0 & \text{if } x \le m \\ \frac{x-m}{n-m} & \text{if } m < x \le n \\ \frac{k-x}{k-n} & \text{if } n < x \le k \\ 0 & \text{if } x \ge k \end{cases} \quad (7)$$

where the starting point m (i.e., the left boundary) denotes the point at which the membership function begins to rise. The peak point n (i.e., the center) is denoted by the point at which the membership function reaches its maximum value, which is typically 1. The ending point, k, is the right boundary at which the membership function returns to zero.

As illustrated in Fig. 7, the membership function of the Residual Energy input is defined by the following configuration:

- **High:** Triangular (50, 100, 100)
- **Medium:** Triangular (25, 50, 75)
- **Low:** Triangular (0, 0, 50)

2. **Application Type:** In the context of Application Type input, the Singleton membership function is a suitable option because it is a categorically distinct variable. Assuming the singleton point m, denote where the membership function is equal to 1. The

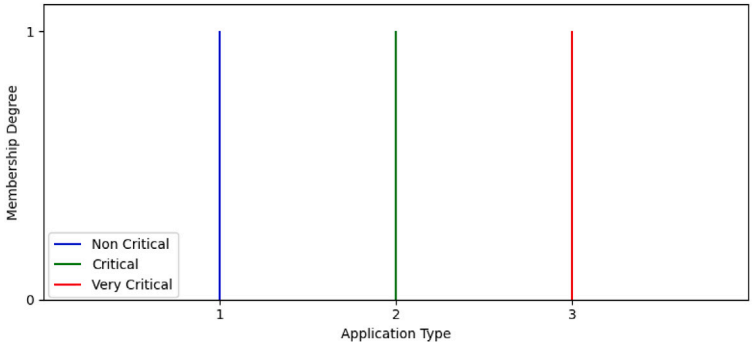

Fig. 8. The Application Type membership function.

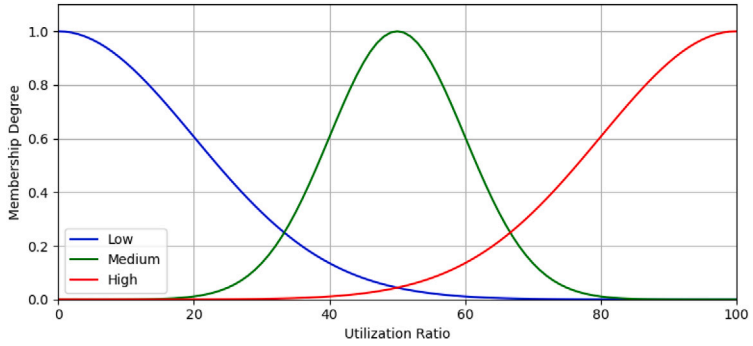

Fig. 9. The Utilization Ratio membership function.

Singleton function is specified by the Formula (8):

$$\mu_A(x) = \begin{cases} 1 & \text{if } x = m \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

As illustrated in Fig. 8, the membership function of the Application Type input is defined by the following configuration:

- **Very Critical:** Singleton (3)
- **Critical:** Singleton (2)
- **Non Critical:** Singleton (1)

3. **Utilization Ratio:** In the context of Utilization Ratio input, the Gaussian membership function is a suitable option as it provides effective transitions between different levels of utilization ratio. This membership function is defined by two parameters: c (the center of the distribution) and σ (the standard deviation), as specified by the Formula (9):

$$\mu(x; c, \sigma) = \exp\left(-\frac{(x-c)^2}{2\sigma^2}\right), \tag{9}$$

As illustrated in Fig. 9, the membership function of the Utilization Ratio input is defined by the following configuration:

- **High:** Gaussian (100, 20)
- **Medium:** Gaussian (50, 10)
- **Low:** Gaussian (0, 20)

4. **Lightweight Hash Functions:** In the context of our output of lightweight hash functions, the Singleton membership function is a suitable option because it is a categorically distinct variable As illustrated in Fig. 10, the membership function of our lightweight hash functions' output is defined by the following configuration:

- **KNOT-512:** Singleton (1), **KNOT-384:** Singleton (2), **QUARK-256:** Singleton (3),
- **SPONGENT-256:** Singleton (4), **KNOT-256:** Singleton (5), **QUARK-176:** Singleton (6),
- **QUARK-136:** Singleton (7), **SPONGENT-128:** Singleton (8), **SPONGENT-80:** Singleton (9).

5.2. Rules evaluation and aggregation

In Fuzzy Logic systems, fuzzy rules are defined as conditional statements that provide the mapping of inputs to outputs. Fuzzy rules enable reasoning with degrees of truth that represent the ambiguity and uncertainty of the actual world, in contrast to classical rules that employ binary logic (i.e., true or false). Each fuzzy rule is composed of two components: an "IF" part and a "THEN" part. This configuration enables the linking of multiple input conditions to a single output action, as demonstrated in Formula (10):

$$\text{IF Input}_1 \text{ is } A_1 \text{ OP Input}_2 \text{ is } A_2 \text{ OP Input}_n \text{ is } A_n \text{ THEN Output is } B, \tag{10}$$

where : $A_1, A_2, \ldots, A_n$ are fuzzy sets describing the inputs, "OP" is a logical operator, and B is the fuzzy set describing the output.

To establish a foundation for the formulation of our rules, a matching process was employed, in which inputs and outputs were aligned according to the following guidelines:

1. For Non Critical IoT applications, we selected lightweight hash algorithms with hash lengths between 80 bits and 136 bits.
2. For Critical IoT applications, we looked at hash lengths between 176 bits and 256 bits.
3. For Very Critical IoT applications, we take hash length equal to or greater than 256 bits.
4. Then, we proceeded to classify lightweight hash functions based on their hash lengths. Lightweight hash functions with greater

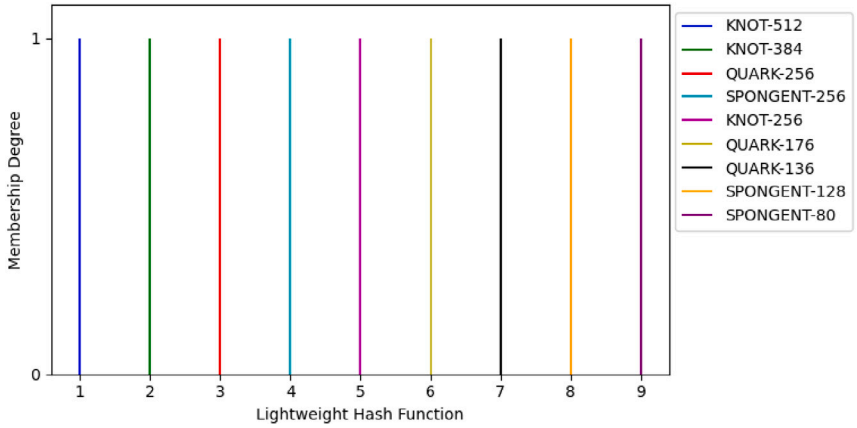

Fig. 10. The lightweight hash functions' membership function.

Table 4
Our fuzzy logic rules.

Application type	Residual energy	Utilization ratio	Lightweight hash function
Very critical	Low	Low	QUARK-256
Very critical	Low	Medium	SPONGENT-256
Very critical	Low	High	KNOT-256
Very critical	Medium	Low	KNOT-384
Very critical	Medium	Medium	QUARK-256
Very critical	Medium	High	SPONGENT-256
Very critical	High	Low	KNOT-512
Very critical	High	Medium	KNOT-384
Very critical	High	High	QUARK-256
Critical	Low	Low	KNOT-256
Critical	Low	Medium	QUARK-176
Critical	Low	High	QUARK-176
Critical	Medium	Low	SPONGENT-256
Critical	Medium	Medium	KNOT-256
Critical	Medium	High	KNOT-256
Critical	High	Low	QUARK-256
Critical	High	Medium	SPONGENT-256
Critical	High	High	SPONGENT-256
Non-Critical	Low	Low	SPONGENT-80
Non-Critical	Low	Medium	SPONGENT-80
Non-Critical	Low	High	SPONGENT-80
Non-Critical	Medium	Low	SPONGENT-128
Non-Critical	Medium	Medium	SPONGENT-128
Non-Critical	Medium	High	SPONGENT-128
Non-Critical	High	Low	QUARK-136
Non-Critical	High	Medium	QUARK-136
Non-Critical	High	High	QUARK-136

length are associated with high residual energy and low utilization ratio in IoT objects.

Table 4 enumerates the sets that describe our fuzzy inputs and outputs, which will be used to define our fuzzy rules using formula (11):

$$\begin{aligned}\text{IF } &\textbf{ApplicationType} \text{ is } (A_1) \text{ AND } \textbf{ResidualEnergy} \text{ is } (A_2)\\ &\text{AND } \textbf{UtilizationRatio} \text{ is } (A_3)\\ &\text{THEN } \textbf{LightweightHashFunction} \text{ is } B,\end{aligned} \tag{11}$$

After determining the fuzzy rules, the Fuzzy Logic system employs fuzzy logic operators (AND, OR, NOT) to evaluate the fuzzy rules as follows:

1. AND operator: $\mu_{\text{Rule}}(\text{Input}_1, \text{Input}_2) = \min(\mu_{A_1}(\text{Input}_1), \mu_{A_2}(\text{Input}_2))$
2. OR operator: $\mu_{\text{Rule}}(\text{Input}_1, \text{Input}_2) = \max(\mu_{A_1}(\text{Input}_1), \mu_{A_2}(\text{Input}_2))$
3. NoT operator: $\mu_{\text{Rule}}(\text{Input}_1) = 1 - \mu_{A_1}(\text{Input}_1)$

Finally, the fuzzy output results obtained from each rule are combined into a single fuzzy output. This process is referred to as the Aggregation phase. This step involves calculating of the union of all the fuzzy outputs. Mathematically, if there are n rules, the aggregated output fuzzy set $\mu_B(\text{Output})$ is defined using the following formula (12):

$$\mu_B(\text{Output}) = \max(\mu_{B_1}(\text{Output}), \mu_{B_2}(\text{Output}), \ldots, \mu_{B_n}(\text{Output})) \tag{12}$$

5.3. Defuzzification

The process of converting the fuzzy output into an exact value output is referred to as Defuzzification. The most common approach to this task is the centroid (center of gravity) technique. Mathematically, if $\mu_B(\text{Output})$ is the aggregated output of the fuzzy set B, the centroid is defined by the formula (13):

$$\text{centroid} = \frac{\int \text{Output} \cdot \mu_B(\text{Output})\, d\text{Output}}{\int \mu_B(\text{Output})\, d\text{Output}}, \tag{13}$$

Defuzzification is simple when dealing with singleton output, where the output fuzzy sets correspond to discrete values. Therefore, rather than using techniques like centroid, we may simply select the output with the highest degree of aggregated membership function μ_B (Output).

6. Performance evaluation

This section describes the simulation environment and parameters used to evaluate our IoT security self-management framework. In addition, the performance of our proposed framework is evaluated by considering the lifetime of the IoT system, energy consumption, the security performance of lightweight hash functions, and the efficiency of attestation.

6.1. Simulation environment

The simulations were performed using the OMNeT++ platform [19], extended with the INET framework [20]. As shown in Fig. 11, we extended the INET framework by integrating two new modules, which we enhanced with additional functionalities to support the evaluation of our new solution concerning IoT object integrity. Indeed, the first module focused initially on Lightweight Cryptography (Block Ciphers (LBCs) and Lightweight Hash Functions (LHFs)). While the second module handles the energy consumption of LBCs and LHFs. In addition, it handles the memory usage (RAM) for both ciphers and hash functions, as these functionalities were not initially part of the INET framework. In the new Lightweight Cryptography module, the LBCs and LHFs are implemented using the Crypto++ C++ library [21]. For

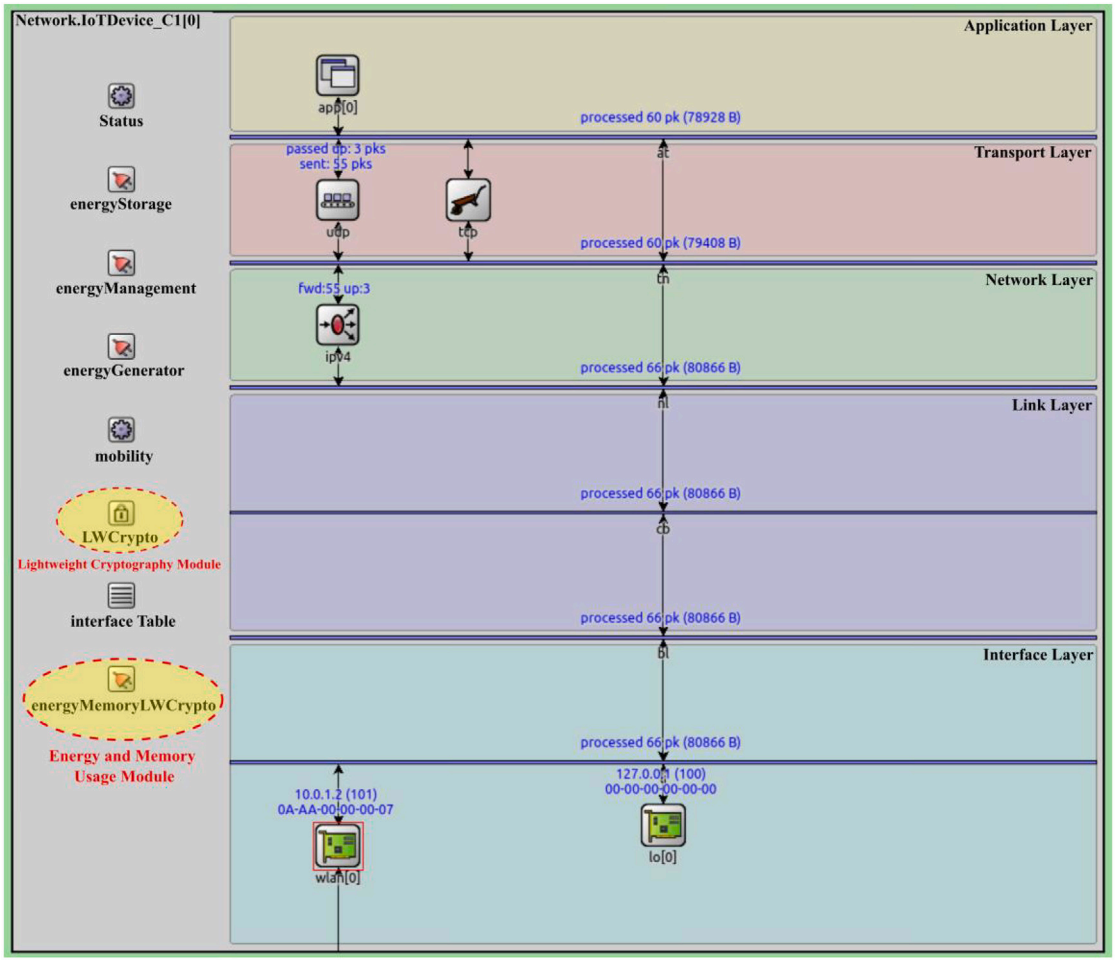

Fig. 11. Extended INET OMNeT++ framework for IoT object integrity evaluation.

Table 5
Autonomic IoT object integrity framework simulation parameters.

Parameter	Value
Topology dimension	1000 m*500 m
Number of clusters	3
Number of IoT devices	15
Battery capacity	2664 J (200 mAh)
Packet length	1024 B
Packet send interval	30 ms
Wireless connection used	IEEE 802.11ac
Wired connection used	Ethernet

the Energy and Memory Usage module, we utilized data from studies conducted in [22,30] to examine the energy and memory consumption of encryption and decryption processes, as well as lightweight hash functions.

Our simulations were conducted on a machine equipped with an Intel Xeon W-2255 3.70 GHz CPU and 64 GB RAM. The default parameters used in our network simulations, enabling the evaluation of our IoT object integrity-based contribution, are summarized in Table 5.

6.2. Performance metrics

Our proposed framework focuses on the self-management of IoT object integrity and aims to reduce energy consumption, extend network lifetime, and minimize the number of attestation process executions. To this end, we specify four key performance metrics for the evaluation of this integrity-related contribution:

1. **IoT Lifetime:** refers to the period during which the IoT system remains operational, which means that at least one IoT device has a battery with more energy left in it than zero.
2. **IoT Energy Consumption:** refers to the energy that remains in the battery of an IoT device after a specific amount of data has been transmitted to security gateways.
3. **IoT Security Performance:** in the context of this study, it refers to the efficiency and effectiveness of our frequently changing remote attestation solution. This metric evaluates the resilience of the scheme against attacks, the impact of our solution on the integrity of the IoT object, and its ability to prevent and respond to security threats.
4. **Attestation Efficiency:** refers to the framework's efficacy in eliminating redundant attestations without compromising security or integrity verification. It is imperative to minimize attestation executions to optimize energy consumption and enhance system efficiency, consequently prolonging the lifespan of the IoT systems.

6.3. Evaluation scenarios and results

6.3.1. Evaluation scenarios

To evaluate the effectiveness of our proposed self-management IoT object integrity framework in terms of attestation time, energy optimization, efficiency of the attestation process, and security performance, we have identified four different evaluation scenarios, which will be referred to as Sc1, Sc2, Sc3, and Sc4, respectively, as described below:

- **Evaluation Scenario 1 (Sc1):** in this scenario, each IoT object uses only one lightweight hash function with the lowest energy consumption for its respective IoT application type. Specifically, for IoT objects implementing the Non-Critical IoT application type, the SPONGENT-80 hash function is used. The QUARK-176 hash function is used for Critical IoT applications, and the SPONGENT-256 hash function is used for Very Critical IoT applications (see Tables 4 and 3). In this scenario, the attestation mechanism is performed for all IoT objects whenever the attestation time is reached, without any IoT object selection process.
- **Evaluation Scenario 2 (Sc2):** in this scenario, IoT objects use the hash function with the highest energy consumption for their respective IoT application types. Specifically, for IoT objects implementing the Non Critical IoT applications, the QUARK-136 hash function is used. For Critical IoT applications, KNOT-256 is used, and for Very Critical IoT applications, KNOT-512 is used (see Tables 4 and 3). Similar to Sc1, the attestation mechanism is performed for all IoT objects whenever the attestation time is reached, without any IoT object selection process.
- **Evaluation Scenario 3 (Sc3):** in this scenario, our Fuzzy Logic-based self-management solution is implemented thanks to the MAPE-K_1 closed control loop within our proposed framework. Instead of using a single hash function, IoT objects dynamically switch between multiple lightweight hash functions according to their state changes (i.e., Application Type, Residual Energy, and Utilization Ratio) as determined by the proposed Fuzzy Logic system. Specifically, for IoT objects running Very Critical applications, the system selects a hash function with digest length equal to or greater than 256 bits, for Critical applications, it selects from hash functions with digest length between 176 and 256 bits, and for Non Critical applications, it selects from hash functions with digest length between 80 and 136 bits (see Tables 4 and 3). As in Sc1 and Sc2, attestation is performed for all IoT objects each time the attestation interval is reached, without requiring the selection of specific IoT objects for attestation.
- **Evaluation Scenario 4 (Sc4):** unlike the previous scenarios, in Sc4, our DBSCAN-based self-management solution is implemented thanks to the MAPE-K_1 closed control loop. This scenario introduces an IoT object selection step. The DBSCAN model autonomously decides whether an IoT object requires attestation verification or not, depending on the anomaly detection in the object's state (i.e., Packet Rate, Energy Consumption, Utilization Ratio, Memory Usage, Residual Energy, and Application Type). In addition, as in Sc3, a dynamic hash function switching approach based on a Fuzzy Logic system is used, with a set of lightweight hash functions instead of a single one. This scenario is implementing the complete proposed IoT objects integrity self-management framework.

As shown in Fig. 12, we simulate each scenario using an IoT architecture consisting of three clusters. Each cluster has a Low-Level Security Gateway (LL-SG) wirelessly connected (via IEEE 802.11ac) to a set of IoT devices that have the same application type (NC, C, or VC) and different operating modes: Always On, Fixed On/Off, or Dynamic On/Off periods (as explained in Section 4.1). All these LL-SGs are connected via wired links to a High-Level Autonomic Security Gateway (HL-ASG).

We use identical application domains and common packet traffic payloads (1024 bytes) to ensure a fair comparison across all scenarios. The on/off phases for the IoT objects' operating mode are replicated across all clusters to ensure a fair comparison. In addition, all scenarios use the same type of IoT device: a Raspberry Pi 3 powered by a 3.7 V, 200 mAh battery providing 2664 joules (J) of power.

6.3.2. Evaluation results

To evaluate our IoT object integrity self-management framework, we conducted two types of simulations based on the four scenarios outlined in Section 6.3.1. The first set of simulations runs until the batteries of the IoT objects are exhausted. This approach allows us to compare the IoT system lifetime, security performance, and attestation efficiency across the four scenarios (Sc1, Sc2, Sc3, and Sc4). For the second set of simulations, we fixed a data size of 2 GB to be sent by each IoT object. This set of simulations measures the energy consumption for each scenario (Sc1, Sc2, Sc3, and Sc4) across different IoT application types (NC, C, and VC).

The results of the first set of simulations, as shown in Fig. 13, illustrate the evaluation of the global IoT system lifetime across scenarios Sc1, Sc2, Sc3, and Sc4. These results show that the implementation of our DBSCAN fuzzy-based autonomic attestation method (Sc4), which identifies suspicious devices using a DBSCAN model and then attests them with changing lightweight hash functions through a fuzzy logic system, significantly improves the overall IoT system lifetime across all three application types (NC, C, and VC) compared to scenario Sc2 and Sc3, where no IoT object selection step is implemented before attestation verification. The results also show that the frequent change of lightweight hash functions in Sc3 results in a higher lifetime compared to Sc2, where a single hash function is used throughout, highlighting the importance of lightweight hash function change in extending the IoT system lifetime.

In the Sc2 scenario, security is prioritized by selecting the most secure lightweight hash functions for each IoT application type (NC, C, and VC), but energy efficiency is not considered. Indeed, higher security hash functions tend to consume more energy, which explains the shorter lifetime of Sc2 compared to other scenarios. However, the IoT system lifetime in Sc4 is lower than in Sc1, where energy efficiency is prioritized over security. In Sc1, IoT objects use a single lightweight hash function with lower energy consumption without considering whether the chosen lightweight hash function is the optimal choice in terms of security for NC, C, and VC IoT applications, sacrificing security for a longer system lifetime. In contrast, our proposed solution in Sc4 strikes a balance between security and energy efficiency, offering enhanced security by modifying hash functions in each attestation iteration while maintaining an acceptable system lifetime.

Concerning the evaluation of energy consumption using the second set of simulations, which involves the transmission of a 2 GB IoT data size across the four security scenarios (Sc1, Sc2, Sc3, and Sc4) and for different IoT application types (NC, C, and VC), the results are shown in Fig. 14. Our proposed solution (Sc4), which implements IoT object integrity self-management by identifying suspicious devices through a DBSCAN model and then attesting them with dynamically changing lightweight hash functions through a fuzzy logic system, performs better in terms of energy efficiency compared to Sc2 and Sc3, where no IoT object selection is performed before attestation verification. Furthermore, the frequent changing of lightweight hash functions in Sc3 does result in energy gains compared to Sc2, where only one hash function is used throughout the simulation, highlighting the role of changing hash functions in energy optimization.

In addition, while Sc4 has a slightly lower energy efficiency than Sc1, the self-management approach in Sc4 provides a much higher level of security for all traffic types (NC, C, and VC) by frequently changing both the hash function and the shared key during the attestation process. The difference in energy efficiency is explained by the fact that Sc1 focuses solely on minimizing energy consumption without considering security, while Sc4 balances energy efficiency and security improvements.

Regarding the security performance metric evaluation of our proposed framework, we specified several graphs that track the use of lightweight hash functions over time (represented by lines) and the generation of shared keys (represented by dots) to ensure the integrity

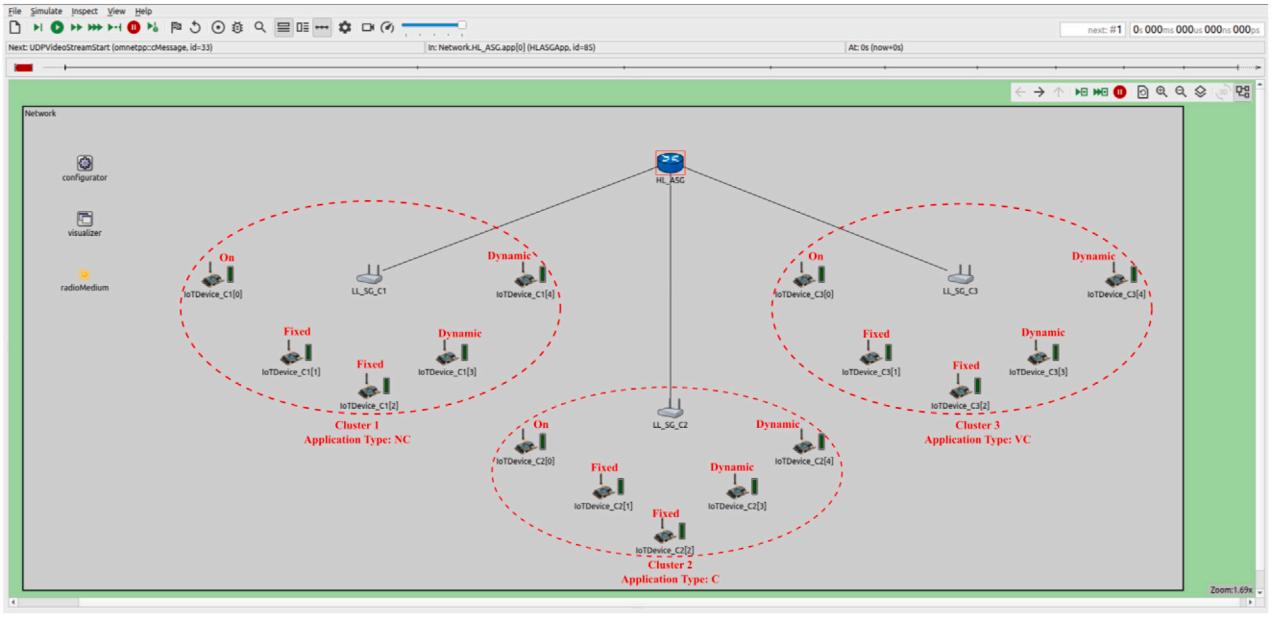

Fig. 12. Simulated IoT architecture for the IoT object integrity framework.

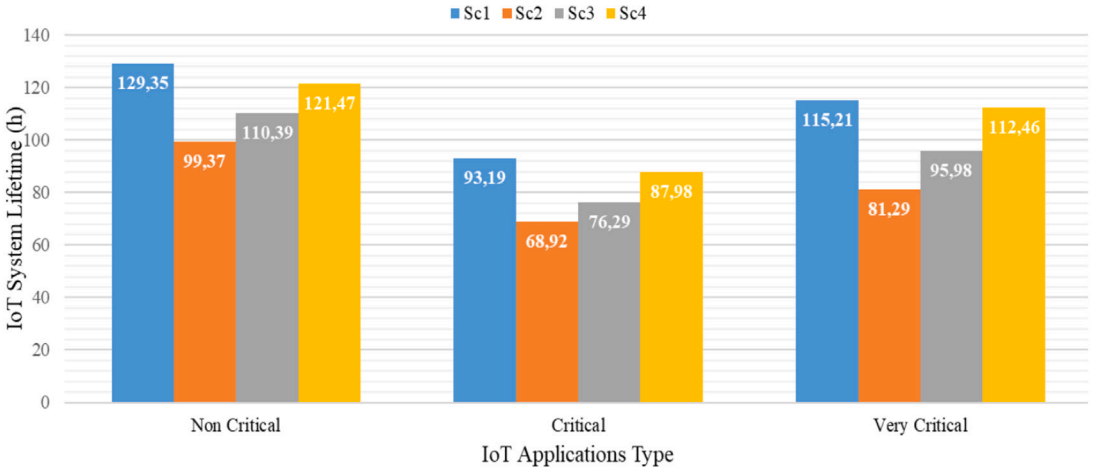

Fig. 13. IoT system lifetime evaluation for NC, C, and VC IoT application types.

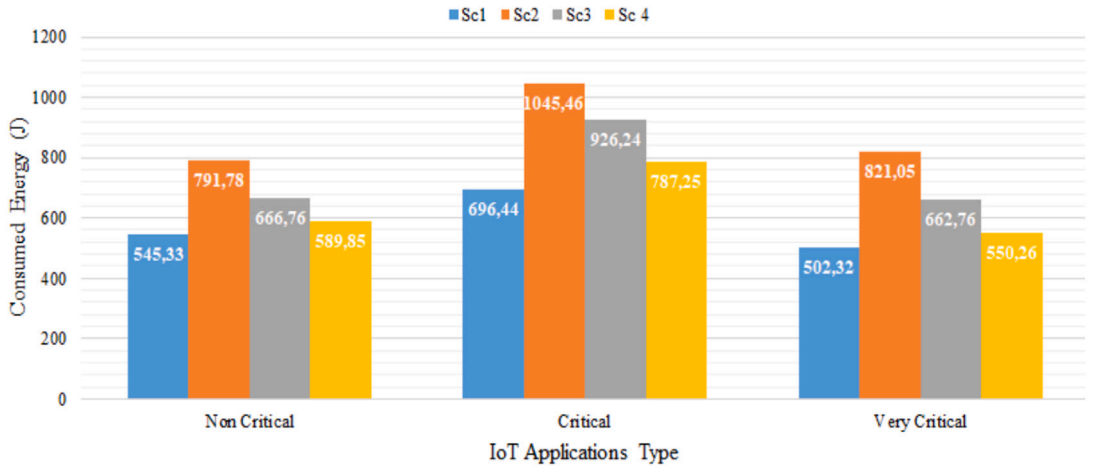

Fig. 14. Consumed energy evaluation for NC, C, and VC IoT application types.

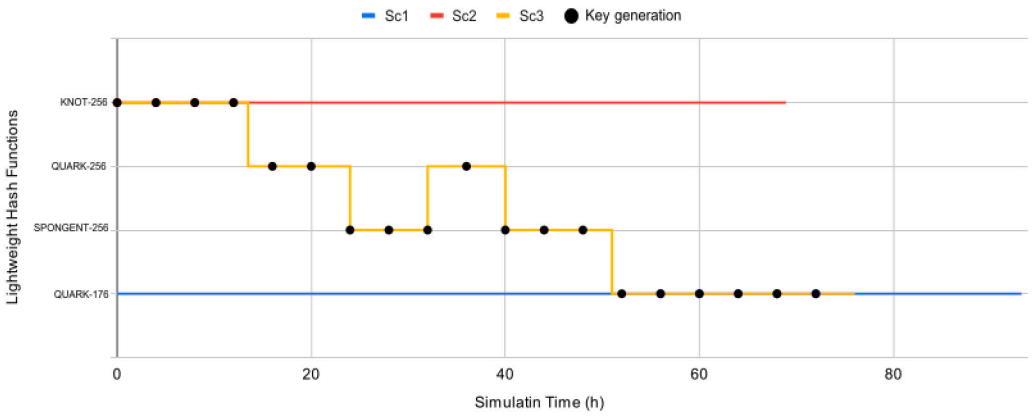

Fig. 15. Evolution of lightweight hash function usage and key generation for NC IoT application.

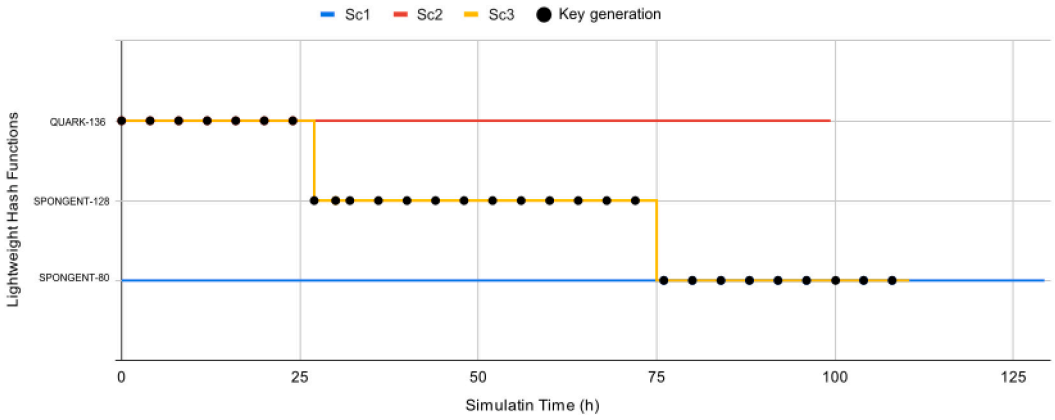

Fig. 16. Evolution of lightweight hash function usage and key generation for a Critical IoT application.

of IoT objects for NC, C, and VC IoT application types. This metric, along with its corresponding graphs, focuses on evaluating the Fuzzy Logic-based part of our self-management approach, which aims to autonomously adjust the lightweight hash function in response to changes in the IoT object states

Fig. 15 illustrates the case of the Non Critical application type in scenarios Sc1, Sc2, and Sc3. Unlike Sc1 and Sc2, which only use the lightweight hash functions SPONGENT-80 and QUARK-136, respectively, our fuzzy-based dynamic hash changing approach, implemented in scenario Sc3, periodically changes between several lightweight hash functions based on the state of the IoT object. These functions include SPONGENT-80, SPONGENT-128, and QUARK-136. In addition, the shared key is frequently updated throughout the simulation in Sc3 to enhance the security of the attestation process.

Furthermore, Sc2 and our fuzzy-based approach (i.e., Sc3) offer a higher level of security than Sc1, which only uses a low-energy, lightweight hash function, SPONGENT-80, without considering security concerns. In addition, the flexible, fuzzy-based self-management techniques implemented in Sc3, which adjust security based on the state changes of IoT objects, provide stronger security and higher resilience against attacks. This is achieved by frequently changing lightweight hash functions and shared key values, thereby reducing vulnerabilities and mitigating the impact of potential attacks compared to traditional methods, such as Sc1 and Sc2, which rely on a single hash function for the entire simulation duration.

As concerning the Critical IoT application type shown in Fig. 16, scenarios Sc1 and Sc2 rely on a single lightweight hash function for the attestation process (QUARK-176 and KNOT-256, respectively). In contrast, our proposed fuzzy-based self-management solution (Sc3) uses multiple hash functions that dynamically change according to the state of the IoT object. This adaptive approach includes algorithms such as KNOT-256, QUARK-176, SPONGENT-256, and QUARK-256. As with the Non Critical application case, our fuzzy-based approach (Sc3) provides superior security compared to Sc1 and Sc2 by frequently changing the lightweight hash functions and generating multiple key values thanks to the session key generation management.

For the Very Critical IoT application type shown in Fig. 17, Sc3 dynamically adjusts the lightweight hash functions based on the changing states of the IoT objects. This includes the use of algorithms such as QUARK-256, SPONGENT-256, KNOT-256, and KNOT-384. In contrast, Sc1 and Sc2 use a single hash function throughout the simulation, with Sc1 using SPONGENT-256 and Sc2 relying on KNOT-512.

After evaluating the security performance of our proposed solution, we have assessed the attestation efficiency. This evaluation focuses on analyzing the minimization of unnecessary attestation processes within our IoT object integrity self-management solution (Sc4), which includes an IoT object selection step based on the DBSCAN model, compared to the other scenarios (Sc1, Sc2, and Sc3), which do not include any IoT object selection process before attestation.

To illustrate this, we present a graph showing the average number of attestation verifications over time for all IoT objects within the same cluster representing the same IoT application type (NC, C, and VC). Additionally, we track the number of attestation verifications performed before detecting a compromised IoT object.

As shown in Fig. 18, which tracks the number of attestation verifications, our IoT integrity self-management approach (Sc4), which determines whether or not an IoT object requires attestation before proceeding with verification based on the DBSCAN model, is compared

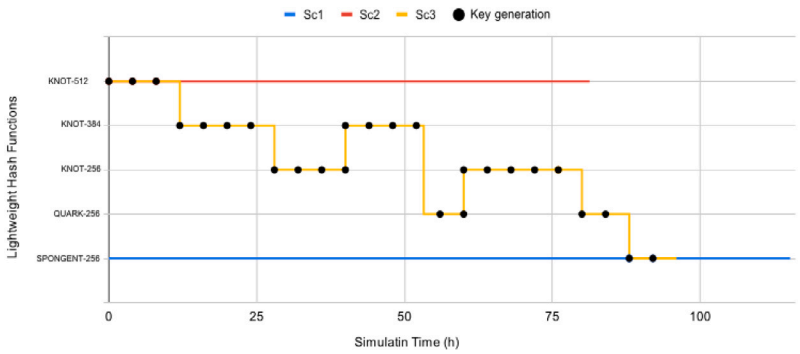

Fig. 17. Evolution of lightweight hash function usage and key generation for VC IoT application.

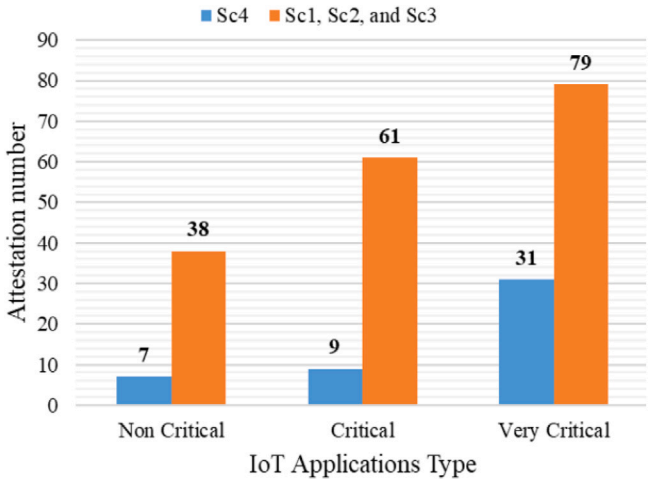

Fig. 18. Average attestation number for NC, C, and VC application types.

to the other scenarios (Sc1, Sc2, and Sc3), which perform attestation without requiring any prior selection of IoT objects. The results show that our approach (Sc4) significantly reduces the number of attestation verifications to less than half compared to the other scenarios that do not implement IoT object selection (i.e., Sc1, Sc2, and Sc3).

In addition, after executing the attack, our method detects the compromised IoT object only after a cumulative number of one attestation check, whereas the other scenarios require a cumulative number of eight unnecessary checks before detection. This demonstrates a significant reduction in redundant integrity checks, resulting in improved energy efficiency and extended system lifetime without compromising the security of the IoT system.

Several key insights can be drawn from these observations. First, our proposed DBSCAN fuzzy-based self-management IoT integrity solution adapts security measures based on the context and domain of IoT applications, ensuring customized protection. Second, across all IoT application types, the proposed solution automatically changes hash functions via Fuzzy Logic and a frequently updated shared key, thanks to session key generation management, based on the state changes of IoT objects, thereby enhancing security and mitigating potential attacks. Third, our self-managing security approach, which identifies IoT objects requiring attestation based on a DBSCAN model and then attests them via an autonomously changing hash function and shared key, significantly emphasizes energy optimization and IoT system lifetime. Finally, our autonomic approach significantly reduces redundant integrity checks, resulting in improved energy efficiency and extended system lifetime while maintaining strong security for the IoT system.

6.4. Security analysis

In this section, we analyze the security of our proposed IoT object integrity self-management framework using a threat model to evaluate its ability to resist various attacks and meet security requirements.

6.4.1. Threat model

- **Code Injection Attacks:** it is a software-based attack where adversaries replace or inject malicious code into the IoT device. The malicious code can change the device's behavior or compromise private information [31].
- **Verifier-Impersonation DoS Attack:** in this attack, the adversary pretends to be the verifier and frequently sends fake attestation requests. This forces IoT devices to repeatedly perform attestation, which can prevent them from performing everyday tasks, potentially leading to a Denial of Service (DoS) attack [1].
- **Transient and Self-Relocating Malware:** in this attack, the adversaries move the location of the malware during the attestation process, hiding it in areas that have already been verified or are not being attested. By moving between attestation intervals, it can evade the security mechanisms designed to detect it [1].
- **Time-of-Check Time-of-Use (TOCTOU) Attack:** in a TOCTOU attack, the attackers take advantage of the time difference between check (attestation) and use of the device. By removing or altering the malicious code between two attestation cycles, attackers ensure that the malware is not detected by the attestation process [11].

6.4.2. Attacks resistance

Assuming an adversary attempts a code injection attack, where the attacker's goal is to compromise one or more devices in the IoT system without being detected by the verifier, our autonomic attestation scheme is designed to verify the integrity of all devices in the IoT system. The solution remains resilient to such attacks because the verifier (i.e., the HL-ASG) accepts and validates an IoT device only if the HMAC calculated by the device matches the one calculated by the HL-ASG. The HMAC incorporates a randomly generated nonce, an updated shared key, and a frequently changing lightweight hash function. This combination prevents the adversary from using an old, valid HMAC result of a legitimate IoT object to bypass the verifier.

Regarding verifier-impersonation DoS attacks, traditional attestation schemes employ a counter to validate the attestation request from the verifier. However, in our solution, the IoT device itself initiates the attestation process, rather than the verifier (HL-ASG), as in the case of challenge- response attestation. This process is undertaken by a secure, read-only clock that triggers the selection phase of our autonomic attestation scheme. If anomalies are detected during the selection phase, the IoT device proceeds with the attestation verification. This design makes our system inherently resilient to verifier-impersonation DoS attacks.

Regarding transient and self-relocating malware, as well as Time-of-Check Time-of-Use (TOCTOU) attacks, research works presented in [11, 32] have demonstrated that both types of attacks rely on knowing the exact attestation time in advance. Unlike fixed periodic attestation intervals that attackers can predict, our self-management approach uses a secure clock to trigger only the IoT object selection process. In contrast, the attestation process itself is performed when the device is selected for attestation by the DBSCAN model. Thus, our self-management solution can significantly increase the difficulty of predicting the attestation time and reduce the possibility of these attacks.

Finally, we can conclude that our proposed IoT integrity self-management approach not only extends the lifetime of IoT systems and optimizes energy consumption by reducing redundant integrity checks, but also establishes a robust security model. This is achieved through two steps: IoT object selection step using DBSCAN and a dynamic attestation step using autonomic fuzzy-based updates of lightweight hash functions and dynamically changing shared keys.

7. Discussion

Our proposed framework enhances IoT security by introducing a lightweight, energy-aware, and autonomic approach to remote attestation tailored to IoT environments. Unlike traditional static or periodic attestation mechanisms, our solution uses adaptive decision-making with DBSCAN and fuzzy logic, as well as a lightweight HMAC solution for attestation verification, to improve security and resource efficiency. Additionally, it opens new avenues for research in adaptive security management that could involve more security services, while emphasizing the need to balance security and energy efficiency in large IoT systems.

Despite its novelty and benefits, this study has limitations. First, the evaluation of the proposed framework relies on simulation due to the lack of publicly available datasets and limited access to large-scale IoT devices. Although it is designed to run under realistic simulations, real-world deployment may introduce additional challenges. Second, this framework assumes the existence of a secure hardware clock in IoT devices that can be used for attestation. However, this assumption may not apply to all IoT objects on the market. This could impact the deployment of the proposed solution. Third, it assumes that attackers will not be able to manipulate the DBSCAN model and fuzzy logic system used for decision-making. The robustness of these systems against manipulation by attackers is an area of interest that has yet to be explored. Finally, the current work primarily focuses on integrity security services. Other security services, such as authentication and access control, are not addressed but could be included in possible extensions of the proposed framework.

8. Conclusions and future works

In this research work, we present a novel, lightweight, and autonomic IoT remote attestation solution. This approach was designed to guarantee the integrity of IoT objects while providing self-management capability. This solution integrates the Autonomic Computing paradigm into the IoT sensing layer by implementing two distinct MAPE-K closed control loops. The initial control loop determines whether an IoT object requires attestation verification using a DBSCAN model and selects the most suitable lightweight hash function for the attestation phase via a fuzzy logic system. The second closed control loop is responsible for managing the attestation process verification itself. This process relies on lightweight HMACs that incorporate a randomly generated nonce, a frequently changing shared key, and frequently evolving lightweight hash functions. Extensive experimentation using INET and OMNET++ has demonstrated the efficacy of our proposed framework in ensuring the integrity of IoT objects, extending the lifetime of the IoT system, optimizing energy consumption, and eliminating redundant integrity verifications. These experiments also underscore the importance of striking a balance between energy efficiency and security in IoT systems.

In future work, we plan to extend our framework by incorporating a hybrid remote attestation approach, which includes both software and hardware-based attestation methods.

CRediT authorship contribution statement

Abdelhamid Garah: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Nader Mbarek:** Writing – review & editing, Supervision, Project administration, Methodology, Formal analysis. **Sergey Kirgizov:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] B. Kuang, A. Fu, W. Susilo, S. Yu, Y. Gao, A survey of remote attestation in Internet of Things: Attacks, countermeasures, and prospects, Comput. Secur. 112 (2022) 102498.
[2] A. Seshadri, A. Perrig, L. Van Doorn, P. Khosla, SWATT: Software-based attestation for embedded devices, in: IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004, IEEE, 2004, pp. 272–282.
[3] W. Feng, Y. Qin, S. Zhao, D. Feng, AAoT: Lightweight attestation and authentication of low-resource things in IoT and CPS, Comput. Netw. 134 (2018) 167–182.
[4] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. Van Doorn, P. Khosla, Pioneer: Verifying integrity and guaranteeing execution of code on legacy platforms, in: Proceedings of ACM Symposium on Operating Systems Principles, SOSP, Vol. 173, 2005, pp. 10–1145.
[5] W. Xu, X. Zhang, H. Hu, G.-J. Ahn, J.-P. Seifert, Remote attestation with domain-based integrity model and policy analysis, IEEE Trans. Dependable Secur. Comput. 9 (3) (2011) 429–442.
[6] H.R. Ghaeini, M. Chan, R. Bahmani, F. Brasser, L. Garcia, J. Zhou, A.-R. Sadeghi, N.O. Tippenhauer, S. Zonouz, {PAtt}: Physics-based attestation of control systems, in: 22nd International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2019, 2019, pp. 165–180.
[7] J. Noorman, J.V. Bulck, J.T. Mühlberg, F. Piessens, P. Maene, B. Preneel, I. Verbauwhede, J. Götzfried, T. Müller, F. Freiling, Sancus 2.0: A low-cost security architecture for IoT devices, ACM Trans. Priv. Secur. (TOPS) 20 (3) (2017) 1–33.
[8] S. Zeitouni, G. Dessouky, O. Arias, D. Sullivan, A. Ibrahim, Y. Jin, A.-R. Sadeghi, ATRIUM: Runtime attestation resilient under memory attacks, in: 2017 IEEE/ACM International Conference on Computer-Aided Design, ICCAD, IEEE, 2017, pp. 384–391.
[9] K. Eldefrawy, N. Rattanavipanon, G. Tsudik, HYDRA: hybrid design for remote attestation (using a formally verified microkernel), in: Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2017, pp. 99–110.
[10] I.D.O. Nunes, K. Eldefrawy, N. Rattanavipanon, G. Tsudik, {APEX}: A verified architecture for proofs of execution on remote devices under full software compromise, in: 29th USENIX Security Symposium, USENIX Security 20, 2020, pp. 771–788.
[11] I. De Oliveira Nunes, S. Jakkamsetti, N. Rattanavipanon, G. Tsudik, On the TOCTOU problem in remote attestation, in: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2021, pp. 2921–2936.
[12] J. Wang, Y. Wang, A. Li, Y. Xiao, R. Zhang, W. Lou, Y.T. Hou, N. Zhang, {ARI}: Attestation of real-time mission execution integrity, in: 32nd USENIX Security Symposium, USENIX Security 23, 2023, pp. 2761–2778.
[13] P. Lalanda, J.A. McCann, A. Diaconescu, Autonomic Computing: Principles, Design and Implementation, Springer Science & Business Media, 2013.
[14] F.M. De Almeida, A.d.R.L. Ribeiro, E.D. Moreno, An architecture for self-healing in internet of things, in: Proceedings of the UBICOMM, Vol. 89, 2015, pp. 76–81.
[15] C. Lin, H. Khazaei, A. Walenstein, A. Malton, Autonomic security management for IoT smart spaces, ACM Trans. Internet Things 2 (4) (2021) 1–20.
[16] A. Garah, N. Mbarek, S. Kirgizov, Enhancing IoT data confidentiality and energy efficiency through decision tree-based self-management, Internet Things 26 (2024) 101219.

[17] H. Krawczyk, M. Bellare, R. Canetti, HMAC: Keyed-hashing for message authentication, 1997, http://dx.doi.org/10.17487/RFC2104, RFC 2104.
[18] A. Ibrahim, A.-R. Sadeghi, G. Tsudik, S. Zeitouni, Darpa: Device attestation resilient to physical attacks, in: Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, 2016, pp. 171–182.
[19] A. Varga, OMNeT++ Community, OMNeT++ Discrete Event Simulator, 2025, Accecible via: https://omnetpp.org/, (Accessed 10 January 2026).
[20] Z. Böjthe, L. Mészáros, G. Szászkő, R. Hornig, A. Varga, A. Török, OMNeT++ Community, INET Framework, 2025, Accecible via: https://inet.omnetpp.org/, (Accessed 10 January 2026).
[21] W. Dai, Crypto++ library, 2007, Accecible via: https://www.cryptopp.com, (Accessed 06 March 2025).
[22] P. Panahi, C. Bayılmış, U. Çavuşoğlu, S. Kaçar, Performance evaluation of lightweight encryption algorithms for IoT-based applications, Arab. J. Sci. Eng. 46 (2021) 4015–4037.
[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[24] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, Vol. 96, 1996, pp. 226–231.
[25] L. van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (Nov) (2008) 2579–2605.
[26] H. Yin, A. Aryani, S. Petrie, A. Nambissan, A. Astudillo, S. Cao, A rapid review of clustering algorithms, 2024, arXiv preprint arXiv:2401.07389.
[27] W.I.D. Mining, Data mining: Concepts and techniques, Morgan Kaufinann 10 (559–569) (2006) 4.
[28] E.H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, Int. J. Man-Mach. Stud. 7 (1) (1975) 1–13.
[29] M.H. Azam, M.H. Hasan, S. Hassan, S.J. Abdulkadir, Fuzzy type-1 triangular membership function approximation using fuzzy C-means, in: 2020 International Conference on Computational Intelligence, ICCI, IEEE, 2020, pp. 115–120.
[30] S. Windarta, S. Suryadi, K. Ramli, B. Pranggono, T.S. Gunawan, Lightweight cryptographic hash functions: Design trends, comparative study, and future directions, IEEE Access 10 (2022) 82272–82294.
[31] A. Francillon, C. Castelluccia, Code injection attacks on harvard-architecture devices, in: Proceedings of the 15th ACM Conference on Computer and Communications Security, 2008, pp. 15–26.
[32] X. Carpent, K. Eldefrawy, N. Rattanavipanon, G. Tsudik, Temporal consistency of integrity-ensuring computations and applications to embedded systems security, in: Proceedings of the 2018 on Asia Conference on Computer and Communications Security, 2018, pp. 313–327.