ZIRCON: Zero-watermarking-based approach for data integrity and secure provenance in IoT networks

Omair Faraj [a,b,*], David Megías [a], Joaquin Garcia-Alfaro [b]

[a] *Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya (UOC), CYBERCAT-Center for Cybersecurity Research of Catalonia, Barcelona, 08018, Spain*
[b] *SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, Palaiseau, 91120, France*

A R T I C L E I N F O

Keywords:
Data integrity
Data provenance
Internet of Things
Intrusion detection
Cybersecurity
Cryptography
Zero-watermarking

A B S T R A C T

The Internet of Things (IoT) is integrating the Internet and smart devices in almost every domain, such as home automation, e-healthcare systems, vehicular networks, industrial control, and military applications. In these areas, sensory data, which is collected from multiple sources and managed through intermediate processing by multiple nodes, is used for decision-making processes. Ensuring data integrity and keeping track of data provenance are core requirements in such a highly dynamic context, since data provenance is an important tool for the assurance of data trustworthiness. Dealing with such requirements is challenging due to the limited computational and energy resources in IoT networks. This requires addressing several challenges such as processing overhead, secure provenance, bandwidth consumption and storage efficiency. In this paper, we propose Zero-watermarkIng based data pRovenanCe for iOt Networks (ZIRCON), a novel zero-watermarking approach to securely transmit provenance and ensure data integrity of sensor data in an IoT network. In ZIRCON, provenance information is stored in a tamper-proof network database through watermarks, generated at the source node before transmission. We provide an extensive security analysis showing the resilience of our scheme against passive and active attacks. We also compare our scheme with existing works based on performance metrics such as computational time, energy usage, and cost analysis. The results show that ZIRCON is robust against several attacks, lightweight, storage-efficient, and better in energy usage and bandwidth consumption, compared to prior art.

1. Introduction

The IoT is an intelligent system composed of physical objects interconnected in a dynamic network infrastructure, which allows it to collect and exchange data between different sources and destinations [1,2]. These objects route data being captured from the environment to the unsafe Internet to be managed, processed and analyzed using different technologies. This makes it easier for an attacker to access the network and, thus, the system becomes vulnerable to intrusions [3]. Such intelligent systems are being used in various applications, such as healthcare systems, home appliances, car automation, industrial control, or environmental monitoring, among others [4,5]. For these reasons, and for more than two decades, protecting and securing networks and information systems have been delivered through Intrusion Detection System (IDS), a security system that monitors network or system activities for malicious activities or policy violations. It analyzes traffic patterns and alerts administrators of potential security breaches. In the context of IoT networks, an IDS can help protect the network by detecting unauthorized access, unusual traffic patterns, and potential attacks such as data forgery or packet replay. This ensures that the network remains secure and data integrity is maintained. It is difficult to apply traditional protection techniques (like cryptography or signature-based techniques) to IoT networks due to some characteristics such as specific protocol stacks, constrained-resource devices, computational and power capabilities, storage limitations and network standards [6]. At first, processing capabilities and storage limitations of network devices that host IDS algorithms is a critical issue. In conventional networks, IDS agents are deployed by network administrator in intermediate nodes that have high computing capacity [7]. On the other hand, network nodes in IoT environments are resource-constrained. Therefore, finding nodes with the ability to support IDS agents is difficult in such systems.

Another major issue is the network architecture. Specific nodes, such as routers and switches, are responsible for forwarding packets to final destinations that are directly connected to end systems in

* Corresponding author at: SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, Palaiseau, 91120, France.
E-mail address: omair_faraj@telecom-sudparis.eu (O. Faraj).

https://doi.org/10.1016/j.jisa.2024.103840

traditional networks. IoT networks are generally multi-hop. In this case, network nodes forward packets simultaneously and act as end systems. Hence, for the sensed data packets to reach the final destination (e. g., gateway, central processing unit, etc.) it will be forwarded through a path of sensor nodes placed on different light poles [7]. Sharing these data packets through the shared wireless medium make the network to be vulnerable to several types of security attacks, such as data forgery, packet replay, data modification, data insertion, or packet drop attack [8,9].

Consequently, there is a need for a lightweight IDS scheme that maintains data integrity, trustworthiness and the ability to secure provenance to ensure that data is forwarded safely. Data provenance, which documents the history and origin of data, provides a history of the data origin and how it is routed over time. In IoT networks, data provenance is crucial for ensuring data integrity and trustworthiness. By maintaining a detailed history of data, it allows for verification of the data's authenticity and helps trace any alterations or tampering, thereby supporting secure and reliable data communication [10,11]. Most of the previous research on provenance considered studying modeling, collecting and querying data provenance without focusing on its security. Moreover, very few approaches considered provenance in sensor networks. In such networks, there is a set of challenges to deploy provenance solutions. These challenges are: (i) manage processing overhead of each individual node, (ii) efficiently transmit provenance while minimizing the additional bandwidth consumption, and (iii) transmit provenance securely from source to destination with the prompt react to any attack [10].

In this paper, we provide a complete framework called Zero-watermarkIng based data pRovenanCe for iOt Networks (ZIRCON) that deals with the above mentioned provenance challenges while ensuring data integrity in an IoT network. These requirements can be achieved through watermarking techniques that are lightweight and require less computational and storage capabilities. Watermarking techniques involve embedding a hidden and unique identifier into data to protect it from unauthorized copying or tampering. The watermark can be extracted or verified later on to ensure the data's integrity and authenticity. On the other hand, zero-watermarking protects digital content without altering the original data by embedding a unique identifier into the content's metadata, ensuring the data remains unchanged while still allowing for ownership verification. Scalability is also an important requirement, since the size of provenance increases proportionally as the number of nodes engaged in the forwarding process increases. To address this issue, we introduce a tamper-proof network database, connected to all nodes and gateway, that stores the provenance information at each hop. In this paper, we propose a zero-watermarking approach that securely communicates provenance information through a tamper-proof database and provides data integrity for real-time systems. This secure storage system, designed to resist unauthorized modifications, is connected to all nodes and the gateway, storing provenance information at each hop. This ensures that the provenance records cannot be altered. A watermark is generated at the source IoT device from the provenance information (IP address, data packet sensed time or received time, packet sequence number) and a hash value of the data payload. The node stores this watermark in a tamper-proof database and embeds it with the data packet to be sent through transmission channel. At intermediate nodes, the watermark is re-generated from original data for verification procedure. Finally, at the gateway, data integrity is verified through watermark re-generation process and the stored watermarks forming the complete provenance information are queried for validating provenance of data and constructing the data path. Hence, the proposed framework provides secure provenance transmission, scalability, secure transmission of watermarks and an in-route data integrity at each point in the network.

1.1. Contribution and plan of the paper

The main contributions of this paper are summarized below.

- We propose ZIRCON, a novel zero-watermarking scheme for IoT networks to ensure data integrity in single- and multi-hop scenarios.
- We use ZIRCON as a solution for to securely transmit provenance information of sensor data in IoT networks that is based on a zero-watermarking approach with a tamper-proof network database.
- We analyze the security of our approach under two main adversary models: passive and active.
- We evaluate the performance of our approach with respect to some related techniques reported in the related literature.

The remaining sections of the paper are structured as follows. Section 2 provides some preliminary background and surveys related work. Section 3 presents the proposed zero-watermarking approach and solutions. Section 4 analyzes the security of the proposed scheme. Section 5 describes simulation results and analysis to validate our approach. Section 6 provides discussion and suggests some directions for future research. Finally, Section 7 concludes the paper.

2. Background and related work

Digital watermarking is one of the well known advances in Wireless Sensor Network (WSN) security. It can detect if sensory data have been modified in a precise way, and prevents the interception of this data effectively. Additionally, it can be used for protecting content integrity of multimedia digital works as images, audio and video, and copyright information [12]. Digital watermarking has many advantages over other security techniques such as:

1. The three watermarking processes (generation, embedding and extraction) requires less energy than traditional encryption due to its lightweight calculations.
2. Carrier data directly holds watermark information without adding any network communication overhead [13].
3. Digital-watermarking reduces end-to-end delay (due to lightweight watermark generation process) in a significant way compared to traditional security techniques with high complexity.

There are two main types of digital watermarking: fragile watermarking and robust watermarking (based on anti-attack properties: fragile watermark becomes undetectable after data being modified, while robust watermarks can survive many forms of distortion) [14,15]. Robust watermarking is mainly used for copyright protection and is not sensitive to tampering. On the other hand, fragile watermarking is greatly sensitive to altering, and any change in the carrier leads to a failure in the extraction of watermark, that is used in verification of data integrity [16].

Watermarking algorithms consist mainly from three processes: watermark generation, watermark embedding and watermark extraction and verification. Based on fragile watermarking algorithms, the source nodes collect data and generate a watermark. Then, this watermark is embedded in the original data using a predefined rule to construct a watermarked data packet that will be transferred to the destination node through the network. Through the transmission channel, the packet may be subject to many types of attacks and unauthorized access. The destination node receives the packet to extract the digital watermark and separates the original data based on the defined rule used at the source node. The restored data is then used to re-generate a watermark using the generation algorithm applied at the sensing node. Finally, the extracted watermark and the re-generated watermark will be compared to verify data integrity [17].

Table 1
State-of-the-art data provenance methods, compared to our approach (cf. ZIRCON, last row).

Ref.	Data integrity	Data tracing	Provenance method	Provenance information	Security analysis	Addressed attacks
Kamal and Tariq (2018) [18]	✔	✗	Link Fingerprints	Received Signal Strength Indicator (RSSI)	✗	Data tampering
Sultana et al. (2011) [19]	✗	✔	Watermarking	Node Identifier	✗	Packet dropping
Sultana et al. (2015) [20]	✗	✔	Bloom Filter	Node Identifier and packet sequence number	✔	Eavesdropping, Provenance tampering, Provenance replay
Suhail et al. (2020) [21]	✔	✔	Routing Table	Source and Destination Node Identifier	✔	Packet replay, Provenance tampering, Packet dropping
Liu and Wu (2020) [22]	✗	✔	Common Substring Matching	Sending Node Identifier, packet sequence number, path index and processing record	✗	Node compromise attack
Siddiqui et al. (2019) [23]	✗	✔	Bloom Filter	Source IP Address	✗	Packet replay
Aman et al. (2021) [24]	✔	✗	Wireless Fingerprints	Physical Unclonable Functions (PUFs)	✔	Eavesdropping, Data Tampering, DoS attack, Node compromise attack
ZIRCON, Faraj et al. (2023)	✔	✔	Zero-watermarking	Sending node IP address, timestamp and packet sequence number	✔	Eavesdropping, Packet replay, Packet dropping, Integrity attack, Data Modification, Database authentication attack, Provenance tampering

Table 2
System notation and parameters.

Notation	Description
$d_{n,k}$	Captured data packet k from node n
N	Number of nodes in the network
H	Number of hops the data packet $d_{n,k}$ routed through
w_{ip}	IoT Device n IP Address
w_t	Sensed data (d_n) capturing time
w_{sq}	Sensed data (d_n) generated sequence number in single-hop scenario
w_{sq_i}	Generated by hop sequence number in multi-hop scenario
$sw_{f_{n,k,i}}$	Sub-watermark i generated using data features of $d_{n,k}$
$sw_{h_{n,k,i}}$	Sub-watermark i generated using hash function for $d_{n,k}$
$E(sw_{f_{n,k,i}})$	Encrypted sub-watermark $sw_{f_{n,k,i}}$
$W_{F_{n,k,i}}$	Final generated watermark i of data packet k from node n
$d_{(n,k)W_{F_{n,k,i}}}$	Watermarked data
$R(W_{F_{n,k,i}}$	Re-generated watermark
$R(sw_{f_{n,k}})$	Re-generated sub-watermark from data features
$R(sw_{h_{n,k,i}})$	Re-generated sub-watermark from hash function
$p_{n,k,i}$	Provenance record i of data packet k from node n
$P_{n,k}$	Set of provenance records of data packet k from node n
$\|\|$	Concatenation
$ENC()$	Encryption function
$DEC()$	Decryption function
$H()$	One-way hash function
K_j	jth Generated and distributed secret key for encryption/decryption
$QRY()$	Query function from network database
$STR()$	Store function to network database
I_l	Intermediate node l
S_n	Source node n
G	Base station or Gateway
q	Number of bits deleted by an attacker

In this paper, we focus on securely transmitting captured data and provenance using zero-watermarking approach based on fragile watermarking. The works that are related to the proposed scheme fall into two classes: data integrity using watermarking and provenance security. The notation and their description used in this paper are listed in Table 2. Below, we discuss fragile watermarking-based approaches and provenance security methods.

2.1. Data integrity using digital watermarking in IoT networks

The literature includes relevant watermarking algorithms used for data integrity and secure transmission in WSN and IoT environments. Existing watermarking schemes can be classified into two main methods: regular watermarking schemes and zero-watermarking schemes.

2.1.1. Regular watermarking schemes

To solve the issue of synchronization between sender and receiver nodes in single chaining techniques such as SinGle chaining Watermark (SGW) [25], Lightweight Chained Watermarking (LWC) [26] and lightweight fragile watermarking scheme (FWC-D) [27], a dual-chaining technique is proposed in [28]. It generates and embeds fragile watermarks into data using dynamic groups. A reversible watermarking-based algorithm for data integrity authentication is proposed in [29]. It applies prediction-error expansion for avoiding any loss in sensory data. Every two adjacent data items are grouped together, and the algorithm uses the first one to generate the watermark, and the other as a carrier for the watermark. Sun et al. [30] propose a lossless digital watermarking approach which embeds the generated watermark in the redundant space of data fields. The method does not increase data storage space due to the fixed size of redundant space. However, data integrity is only checked at the base station side. Guoyin et al. [17] propose a watermarking scheme for data integrity based on fragile watermarking in order to solve the problem of resource restrictions in the perception layer of an IoT network. They design a position random watermark (PRW) that calculates the embedding positions for watermarks. The watermarks are generated using the SHA-1 one-way hash function, which is then embedded to the dynamic computed position. This scheme ensures the data integrity at the sink node. It cannot verify integrity along the route of the communication. The drawback of these solutions is that they only provide an end-to-end verification, since watermark generation and verification is based on a group of data packets. Hence, we consider our approach as a better alternative to serve hop by hop data integrity verification between source and final destination of data packets.

Zhou et al. [31] propose a secure data transmission scheme using digital watermarking technique in a WSN. In this scheme, the hash value of two time-adjacent sensitive data is calculated at the source node using a one-way hash function. Then, according to a digital watermarking algorithm, the sensitive data will be embedded into part of the hash sequence as watermark information. The scheme lacks any proof of concept and security analysis to check its robustness against different types of attack. Another solution for attack detection presented in [32]. The method develops a Randomized Watermarking Filtering Scheme (RWFS) for IoT applications by deploying an en-route filtering that removes injected data at early stage communication based on randomly embedding a watermark in the payload of the packet. The scheme encrypts the data packet before transmission, which encounters additional computation overhead for sensor nodes. To minimize energy consumption, Lalem et al. [33] propose a distributed watermarking technique for data integrity in a WSN using linear interpolation for watermark embedding. The technique allows each node to check the integrity of the received data locally by extracting the watermarked data and generating a new watermark for verification. This method is based on a fixed watermark parameter for all sensor nodes in the network that can be vulnerable to many attacks. Soderi et al. [34] propose WBPLSec protocol a watermarked-based approach to enhance the security of communication between nodes. This protocol utilizes a blind watermark algorithm along with a jam receiver operating over an acoustic channel to exchange a 128-bit key with neighboring devices. The process involves modulating the message using the Direct Sequence Spread Spectrum (DSSS) technique and embedding it with a shifting key to create a watermarked segment. This segment is then encoded into a waveform audio file format and transmitted via an amplifier. The receiver can extract the clean code by utilizing the information embedded in the watermark. However, this proposal necessitates the existence of a private and covert channel among nodes, which can lead to a reduction in wireless bandwidth.

2.1.2. Zero-watermarking schemes

Zero-watermarking is a relatively new digital watermarking method. Each watermarking scheme has a different watermark generation, embedding and extraction process such as unique code (embedded in information hiding schemes), changing position of bits and hash functions (cryptographic schemes) [35]. In zero-watermarking schemes, watermarks are generated by source node from the extraction of important data features of original data without amendment to the data of these features. Different generation functions can be applied in zero-watermarking. The generated watermarks are not embedded in the data payload, but it is invisibly integrated in the data packet and the data remain unmodified.

Although several zero-watermarking techniques exist in the related literature, few methods are proposed to protect data integrity in IoT environments. In [36], a secure data aggregation watermarking-based scheme in homogeneous WSN (SDAW) is presented as a new security technique to protect data aggregation. In this mechanism, watermarks are generated using the Medium Access Control (MAC) address of sensor nodes and collected data by a one way hash function (SHA-1). The proposed scheme guarantees secure communication between the aggregation nodes and the base station. However, the authors do not provide any security analysis to check the resistance of this scheme against different type of attacks. To ensure trustworthiness and data integrity in an IoT network, Hameed et al. [35] propose a zero-watermarking technique which generates and constructs a watermark in the original data before being transmitted. The generation process of the watermark is based on the original data features (data length, data occurrence frequency and data capturing time). This scheme is shown to be more computationally efficient and requires less energy compared to cryptographic techniques or reversible watermarking schemes. The proposed approach is vulnerable to modification attack, since it only uses data length, data occurrence frequency and data capturing time to generate the watermark. Hence, if data is modified by changing position of data values, the attack will not be detected.

2.2. Data provenance in IoT networks

The concept of data provenance is used in many fields of study. Each application domain defines provenance in a different way [37]. Data provenance in IoT networks serves to guarantee data trustworthiness by collecting the lineage of ownership and actions executed on collected data from the source node to its ultimate destination. It is essential to record provenance for every data packet acquired from source nodes and trace the involvement of forwarding nodes throughout the data transmission process, but deploying such a solution presents numerous challenges. One significant challenge is the rapid increase in provenance data during the transmission phase in IoT networks. Additionally, limitations arise from the constraints imposed by data storage capabilities, bandwidth, and energy consumption of nodes [38]. Data provenance guarantees that the data received at the final destination is trusted by the user, verifying that the data is captured by the authorized specific IoT node at the stated time and location [39]. Provenance can be represented as a path of nodes from source to destination as shown in Fig. 2.

Several researchers have used the concept of data provenance to identify the origin of data, track the ownership to serve the authenticity of data and assess trustworthiness. Kamal et al. [18] present a lightweight protocol for a multi-hop IoT network to provide security for data and achieve data provenance. The protocol uses link fingerprints generated from the Received Signal Strength Indicator (RSSI) of IoT nodes in the network. Data provenance is achieved by attaching the encoded link fingerprint to the header of data packet at each hop. The packet header is then decoded in sequence at the server. Provenance information grows very fast, which requires transmitting large amount of provenance information with data packets (i. e., increasing the bandwidth overhead). Sultana et al. [19] establish a data provenance mechanism to detect malicious packet dropping attacks. The method relies on the inter-packet timing characteristics after embedding provenance information. It detects the packet loss based on the distribution of the inter-packet delays and then identifies the presence of an attack to finally localize the malicious node or link. In their work, Salmin et al. [20] present a secure provenance scheme for WSN. Their approach involves embedding provenance information into a Bloom filter (BF), which is transmitted alongside the data. This scheme effectively addresses the challenges posed by resource constraints in WSNs. It requires a single channel for data and provenance transmission. The scheme overlooks data integrity and only focuses on studying the packet drop attack. Suhail et al. [21] present an approach called Provenance-enabled Packet Path Tracing for IoT devices connected through the RPL protocol. Their scheme involve incorporating sequence numbers into the routing entries of the forwarding nodes' routing table, establishing a node-level provenance. Additionally, they introduced a system-level provenance that encompassed destination and source node IDs, enabling complete packet trace capture. To retrieve the entire data provenance using this approach, it is essential to sequentially access the storage space of each node along the routing path. Hence, the base station cannot independently decompose the complete provenance of each data packet. Liu et al. [22] introduced an algorithm for compressing provenance called index-based provenance compression. To reduce the overall size of the provenance data, their approach combines the concept of typical substring matching with path identifier and path index to represent path information within data provenance. Additionally, they expand the data provenance scheme to include attack detection and present a method for identifying malicious nodes based on this expanded scheme. The proposed scheme falls short in terms of ensuring data integrity, lacks a thorough security analysis within a defined threat model and results in computationally intensive operations. Siddiqui et al. [23] present a data provenance technique for IoT devices that employs BF and attribute-based encryption. However, this approach poses challenges as IoT devices typically have limited memory capacity, making it impractical to store extensive provenance information. Furthermore, the technique is susceptible to physical attacks, whereby an attacker can easily manipulate the stored provenance information within an IoT device. Aman et al. [24] propose an analytical model to create a mechanism that enables the establishment of data provenance in IoT systems. Their approach incorporates physical unclonable functions (PUFs) and the extraction of fingerprints from the wireless channel, along with the implementation of mutual authentication and anonymity measures, all aimed at achieving robust data provenance. The approach lacks consideration for the multi-hop scenario and fails to adequately address tracking of data packet provenance. Table 1 presents a concise comparison between ZIRCON and other data provenance techniques mentioned previously.

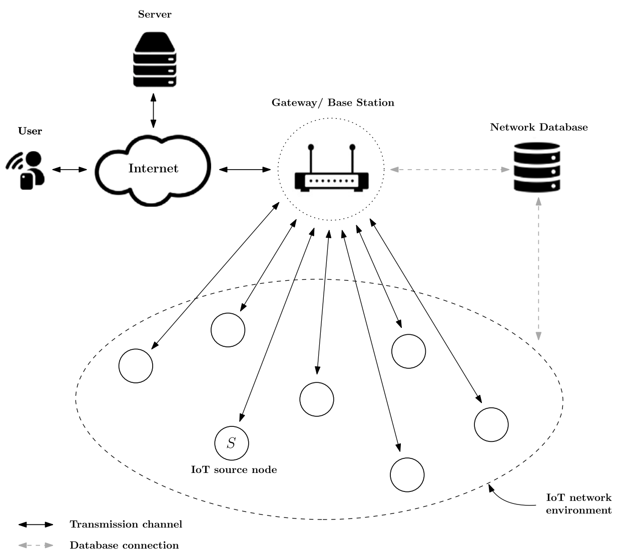


Fig. 1. Single hop network model. The source nodes are directly connected to the base station and to the tamper-proof network database through a transmission channel. The gray dotted two-way arrow represents the store-query connection between the network entities and the database.

3. Proposed system model

In this paper, we propose a zero-watermarking approach to verify data integrity at each hop of an IoT environment (i. e., from source node to gateway). Additionally, we ensure secure provenance of sensory data using a tamper-proof database connected to the gateway and to each node in the network. Data provenance information is used in the watermarking generation process and embedded with captured data packets to be used in the data integrity and secure provenance verification procedure. The system is composed of the following entities:

- **IoT Source Node:** a small sensing device that collects data from surrounding environment. At each node, watermarking generation and embedding processes are applied to each captured data packet. The device performs some operations and communication procedures in the network.
- **IoT Intermediate Node:** an advanced sensing device with more power and computational capabilities, responsible for forwarding data packets from source nodes to the base station. It also performs watermark generation, embedding and storing on the received data packets. Data integrity will be checked for each data packet forwarded at this stage.
- **Base Station or gateway:** receive the forwarded data packets for data processing. Checks data integrity and provenance recovery, and applies the attack detection procedure.
- **Network Database:** a secure network database which is connected to the network nodes and gateway. It stores provenance information at each hop of the data path.

Notice that the suggested coordination approach is consistent with other strategies already in place on IoT-like deployments, such as tasks for uplink interference management, frequency hopping, or frequency allocation, among others, which combine distributed coordination tasks under the control of central entities [40].

3.1. Network model

In the proposed network model, the network is assumed to consist of N IoT devices that are distributed in an IoT network. The network is deployed in an $L \times W$ area. Devices are connected to a gateway or base-station which is the management and controller unit. Nodes are of two types: normal sensor nodes and intermediate sensor nodes. Sensory data is routed from normal source nodes to the gateway through intermediate nodes. This implies that intermediate nodes have m times more energy than normal nodes (i. e., energy of a normal node = E_0, energy of intermediate node = $E_0 + m \times E_0$). Furthermore, a tamper-proof database is connected to the gateway and to each node. It is assumed that the database cannot be compromised by the attacker. The model consists of two main scenarios: single-hop and multi-hop. In the single-hop scenario, IoT devices transmits sensory data directly to the gateway through the transmission channel, as shown in Fig. 1. However, in the multi-hop scenario, sensory data is routed from the source node to the gateway through intermediate nodes as shown in Fig. 2.

3.1.1. Single hop model

In this scenario, the process of data integrity and secure provenance is composed of different units that form the overall system model as described in Fig. 3. Sensor nodes capture data from the surrounding environment and send it to the feature extraction unit. The IP address of the IoT device, data capturing time and a generated unique packet sequence number are extracted and encrypted to generate a sub-watermark in the first sub-watermark generation unit. This sub-watermark forms the provenance record of a particular data packet. A hash function is used to generate another sub-watermark that is concatenated with the first one to generate a final watermark. The generated final watermark is then stored in a tamper-proof database. The data packet is then sent to the gateway through the transmission channel. At the gateway, data is received and forwarded to the zero-watermark re-generation unit. After the re-generation process, the stored watermark is queried from the database to be compared with the re-generated watermark in the watermark verification unit for provenance integrity check. A double verification procedure is applied for both integrity and provenance. At this stage, the gateway detects whether data and provenance is altered or not and performs either attack procedure or validates the origin of data received.

3.1.2. Multi hop model

The watermark generation, embedding, extraction and verification processes of the multi-hop scenario are shown in Fig. 4. In this model, the data capturing time and IP address are extracted from sensed data packets and a generated packet sequence number are used to generate a sub-watermark, which is then encrypted using a secret key and concatenated with a generated hash value of data payload to construct the final watermark. The first sub-watermark or provenance record is stored in the network database and the final watermark is concatenated with the sensed data $d_{n,k}$ to be forwarded to the next intermediate node through the transmission channel. At the next hop node, the watermarked data is received. The watermark is then extracted from the received data packet. The received data is then used to re-generate a new sub-watermark that will be forwarded to the verification unit along with the extracted watermark and a queried provenance record. The intermediate node takes a decision whether an attack is detected or not. Based on this decision, it performs an attack detection procedure or generates the next-hop watermark that undergoes the same procedure of the source node (generation, embedding and storing); it uses new extracted features and provenance information. The watermarked data reaches the final destination (i. e., gateway) through transmission channel. The last embedded watermark is separated from the watermarked data and a final sub-watermark is re-generated. The data integrity unit accepts extracted watermark, re-generated sub-watermark and queried provenance record as input values to check whether data or provenance is modified or not. After that, the gateway performs two procedures based on the verification result: attack detection procedure or provenance validation.

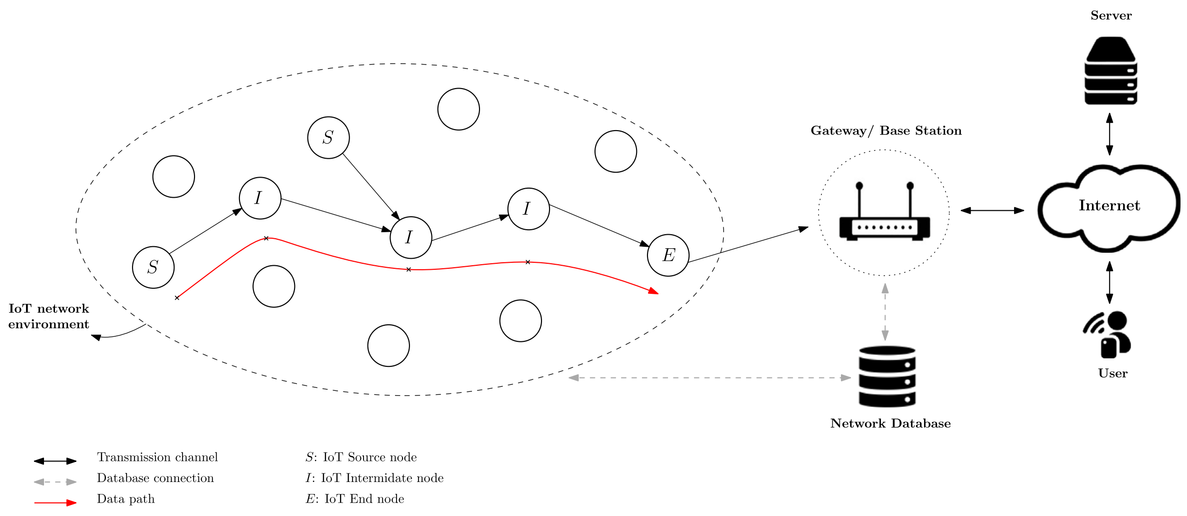


Fig. 2. Multi hop network model. The source nodes are connected to the base station through multiple intermediate nodes. The gray dotted two-way arrow represents the store-query connection between the network entities and the database. The red arrow from the source to the end node represents the complete data path of a particular data packet. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

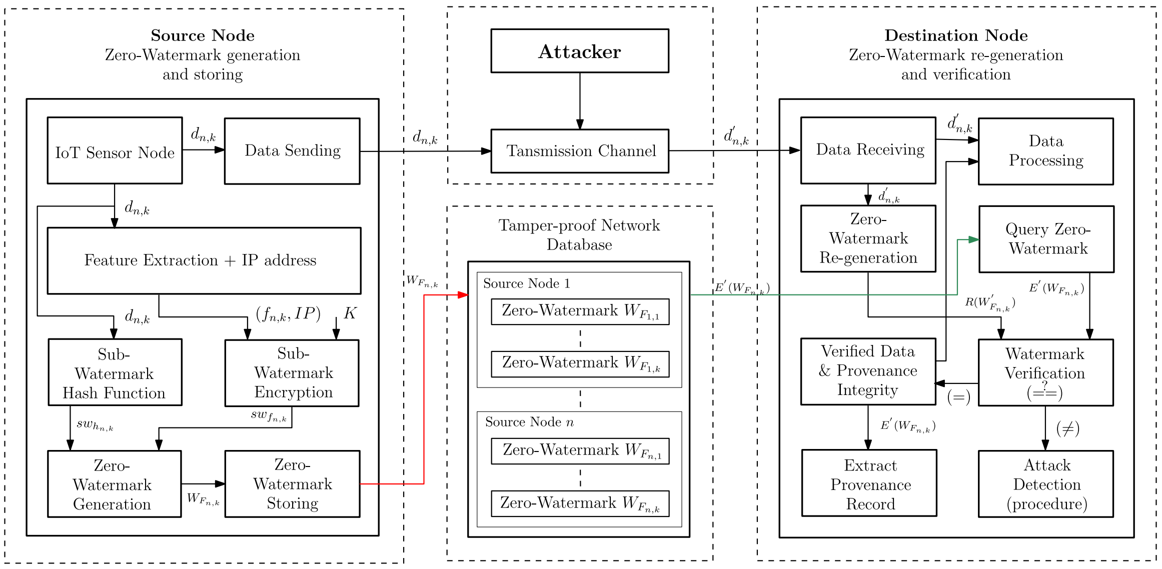

Source Node
Zero-Watermark generation and storing
IoT Sensor Node
Data Sending
Feature Extraction + IP address
Sub-Watermark Hash Function
Sub-Watermark Encryption
Zero-Watermark Generation
Zero-Watermark Storing
Attacker
Tansmission Channel
Tamper-proof Network Database
Source Node 1
Source Node n
Destination Node
Zero-Watermark re-generation and verification
Data Receiving
Data Processing
Zero-Watermark Re-generation
Query Zero-Watermark
Verified Data & Provenance Integrity
Watermark Verification
Extract Provenance Record
Attack Detection (procedure)

Fig. 3. Zero-watermark generation, storing and verification block diagram in single hop scenario. The dotted blocks represent the entities involved in the network. The red arrow represents the *store* function from a source node to the database. The green arrow represents the *query* function from the database to the base station. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.2. Security assumptions

We consider a set of security assumptions for the proposed system as follows:

- Nodes in the network are not trusted entities, i. e., these nodes may be malicious. The protocols provided to guarantee the secure transfer of provenance information that is applied in intermediate nodes and gateway are proven to work properly in the presence of malicious nodes.
- The network database is a trusted and secure entity, it cannot be compromised by an external attacker to access or use its content.
- To allow only authorized gateways to access the network database and query provenance information, only registered gateways are authorized and the query is applied after checking the integrity of the data received from the last hop of the data path.
- The database temporarily stores provenance information of a data packet at each hop from source to destination, only after being proved as trusted data. This linage can be retrieved once (by authorized gateway) and then it is removed from the database.
- The hashing functions used in the system are secure and cannot be inverted.
- The communication of extracted data features and provenance information (sub-watermarks) between source nodes and intermediate nodes, and between intermediate nodes and gateway, must be secure. Provenance information is encrypted using symmetric cryptography and selected data (for integrity check) is hashed using a one-way hash function.
- Symmetric cryptography is restricted to the encryption of short binary strings forming the extracted data feature sub-watermark. Source node, intermediate node and gateway share secret-keys to be used in different steps of the algorithms (encryption/ decryption).
- Secret keys are changed and redistributed after a short random number of watermark generation processes.

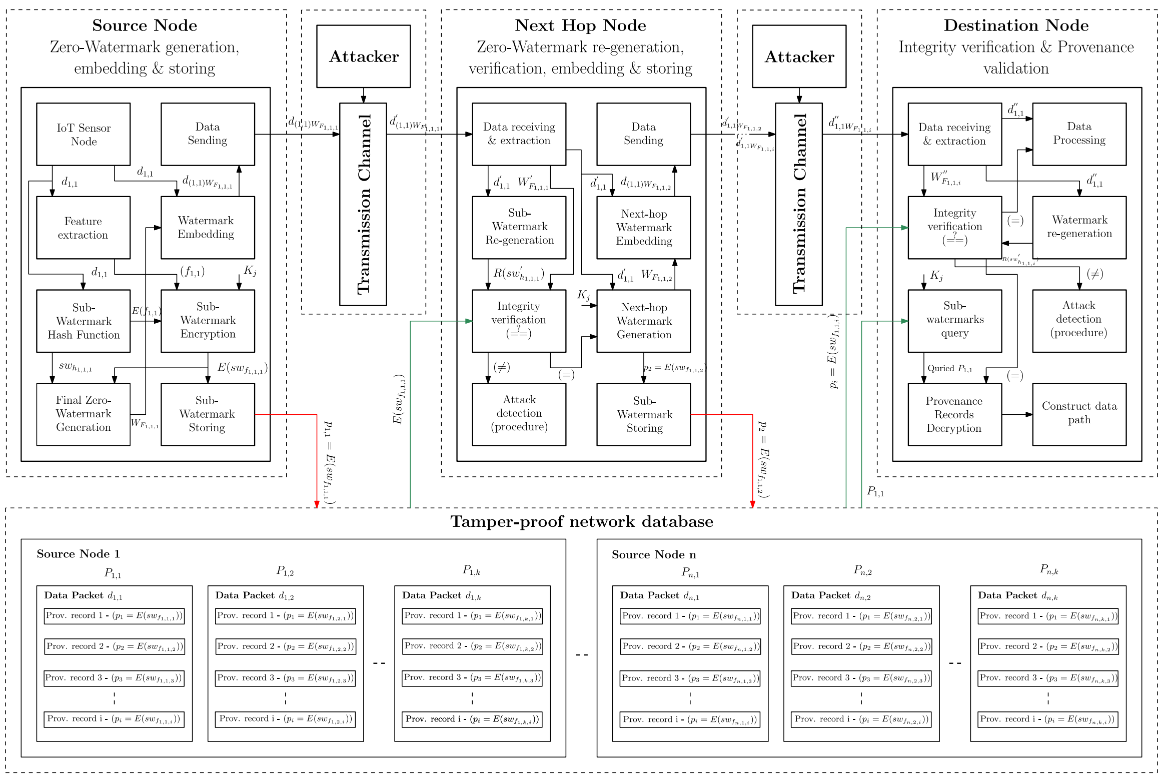

Source Node
Zero-Watermark generation, embedding & storing
IoT Sensor Node
Data Sending
Feature extraction
Watermark Embedding
Sub-Watermark Hash Function
Sub-Watermark Encryption
Final Zero-Watermark Generation
Sub-Watermark Storing
Attacker
Transmission Channel
Next Hop Node
Zero-Watermark re-generation, verification, embedding & storing
Data receiving & extraction
Data Sending
Sub-Watermark Re-generation
Next-hop Watermark Embedding
Integrity verification
Next-hop Watermark Generation
Attack detection (procedure)
Sub-Watermark Storing
Destination Node
Integrity verification & Provenance validation
Data receiving & extraction
Data Processing
Integrity verification
Watermark re-generation
Sub-watermarks query
Attack detection (procedure)
Provenance Records Decryption
Construct data path
Quried $P_{1,1}$
Tamper-proof network database
Source Node 1
Source Node n
Data Packet $d_{1,1}$
Prov. record 1 - ($p_1 = E(sw_{f_{1,1,1}})$)
Prov. record i - ($p_i = E(sw_{f_{1,1,i}})$)
Data Packet $d_{n,k}$

Fig. 4. Zero-watermark generation, storing and verification block diagram in multi-hop scenario. The dotted blocks represent the entities involved in the network. The red arrows represent the *store* function from a source node to the database. The green arrows represent the *query* function from the database to the base station. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- The zero-watermarking method used to embed provenance information is transparent, fragile and secure enough for IoT network applications.

3.3. Threat model

There is a number of different attacks that may be applied against the proposed system. Attack models used to deceive and perform security breaches on different network entities require another party to obtain secret information or access the network database. A threat model similar to the one used in [41] is applied in our scheme. The attacker can perform two types of attacks: passive and active attacks (e. g., external attacks).

1. **Passive attack:** An attacker observes secret information by passively eavesdropping data. The attacker performs an eavesdropping attack that aims to obtain data information through listening to data transmission line in the wireless medium.
2. **Active or external attack:** A malicious attack aiming to destroy information by modifying data packets through launching different kinds of operations. An external or active adversary can launch the following main attacks:

   (a) **Replay attack:** Data packets are captured by an adversary and then resent in the future at a different time interval to deceive intermediate nodes or the gateway.
   (b) **Integrity attack:** An attacker inserts false value(s) into the data packet at the transmission channel to deceive the gateway. Also, the attacker may delete elements of the data packet.
   (c) **Modification attack:** In this attack, data is modified by an attacker without knowledge of the data content.
   (d) **Packet drop attack:** refers to the intentional dropping or discarding of network packets by an attacker. This attack disrupts communication between devices, leading to the loss or interruption of data transmission. The attacker selectively discards packets, targeting specific devices or specific types of data.
   (e) **Database authentication attack:** An attacker aims to detect and retrieve provenance information stored in the network database.
   (f) **Denial of Service (DoS) attack:** Is a malicious attempt to disrupt the normal functioning of IoT devices or services by overwhelming the network with illegitimate traffic, exhausting its resources, and making it unavailable to legitimate users.
   (g) **Man-in-the-Middle (MITM) attack:** MITM attacks involve intercepting communication between IoT devices and altering the data exchanged.

3.4. Proposed algorithms

In this section, a precise algorithmic presentation of ZIRCON to conduct the zero-watermarking scheme is described in details. The interaction between source nodes, intermediate nodes, the gateway, and the tamper-proof network database is described in Fig. 5.

3.4.1. Single hop scenario

In this scenario, two algorithms are proposed: watermark generation and storing, and watermark verification.

1. **Watermark Generation and Storage:** Algorithm 1 describes the process of generating and storing a watermark in a single-hop scenario. It accepts sensed data from the IoT device to produce a final watermark. The algorithm extracts the IP address and the data capturing time from the source node and combines it with a generated unique packet sequence number (seq) to generate a sub-watermark $sw_{f_{n,k}}$ as shown in Lines 2–5 of Algorithm 1.

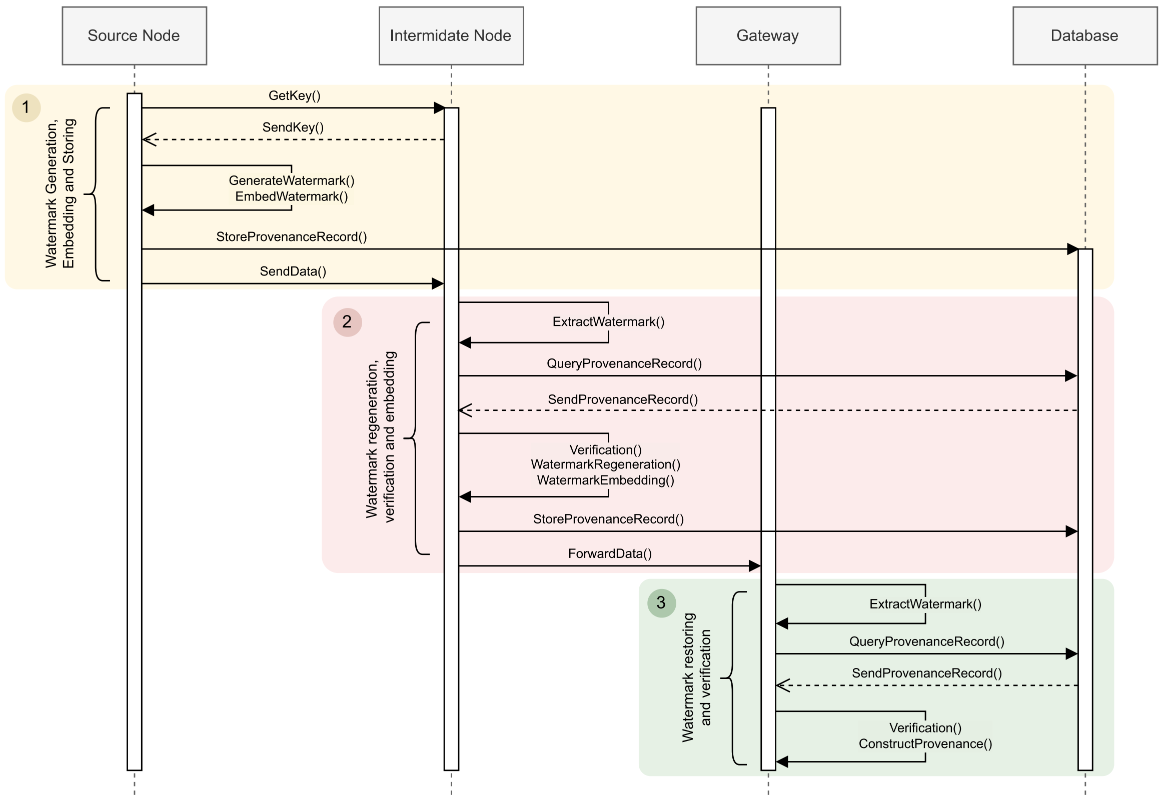


Fig. 5. Sequence Diagram of ZIRCON. The dotted arrows represent a return. Normal arrow represent a message or a request.

Algorithm 1 : Watermark Generation and Storage

```
    input: $d_{n,k}$
    output: $W_{F_{n,k}}$
 1: procedure WATERMARK GENERATION AND STORAGE
 2:     $w_{ip} \leftarrow$ IoT Device $n$ IP Address
 3:     $w_t \leftarrow$ captured data.sensing time $(d_{n,k})$
 4:     $w_{sq} \leftarrow$ packet sequence number $(seq(d_{n,k}))$
 5:     $sw_{f_{n,k}} \leftarrow w_{ip} \mid\mid w_t \mid\mid w_{sq}$
 6:     $p_{n,k} \leftarrow E(sw_{f_{n,k}}) \leftarrow \mathrm{ENC}(K_j, sw_{f_{n,k}})$
 7:     $sw_{h_{n,k}} \leftarrow H(d_{n,k})$          ▷ Select first 8 bytes of hash output
 8:     $W_{F_{n,k}} \leftarrow E(sw_{f_{n,k}}) \mid\mid sw_{h_{n,k}}$
 9:     $STR(W_{F_{n,k}})$
10:     $Send(d_{n,k})$
11: end procedure
```

The sub-watermark is then encrypted using the secret key K_j to obtain a provenance record $p_{n,k} = E(sw_{f_{n,k}}, K_j)$. Another sub-watermark $sw_{h_{n,k}}$ is generated from the hash value of data payload using a one-way hash function. These two generated sub-watermarks are concatenated to form a final watermark $W_{F_{n,k}}$ as:

$$W_{F_{n,k}} = E(w_{ip} \parallel w_t \parallel w_{sq}, K_j) \parallel H(d_{n,k}) = E(sw_{f_{n,k}}, K_j) \parallel sw_{h_{n,k}} \tag{1}$$

where $\parallel$ denotes the concatenation operator, $W_{F_{n,k}}$ $1 \leq n \leq N$, is the final watermark, N is the number of nodes in the network, H is a lightweight (and secure) hash function, and n is the particular node number. The watermark generation algorithm uses the SHA-2 hash function to calculate the hash value. The advantage of using the SHA-2 hash function over other hash algorithms is that SHA-2 has a lightweight feature that uses 65% less memory than other algorithms, such as the MD5 hash function (which has several vulnerability issues), which is needed in resource-constrained networks [42]. After the generation procedure, the final watermark is stored in the network database and the data packet is sent to the base station as shown in Lines 7–9 of Algorithm 1.

2. **Watermark Querying and Verification:** The process of verifying data integrity and validating data provenance in a single-hop scenario is described in Algorithm 2, which takes the received data $d'_{n,k}$ as an input. Then, a re-generation procedure is performed to re-generate the watermark $R(W'_{n,k})$ and the stored watermark $W_{F_{n,k}}$ is queried from the database. A comparison operation is then applied on the re-generated sub-watermark $R(sw'_{h_{n,k}})$ and the queried sub-watermark $sw_{h_{n,k}}$. If the sub-watermarks are the same, data integrity is verified. Then, another comparison operation is performed that compares the re-generated sub-watermark $E(R(sw'_{f_{n,k}}))$ and the second queried sub-watermark $E(sw_{f_{n,k}})$, for provenance integrity check, as shown in Line 11 of Algorithm 2. If these two sub-watermarks are the same, provenance integrity is verified and the provenance record $p_{n,k}$ that contains provenance information is decrypted using the secret key K_j. The IP address, data capturing time and packet sequence number are obtained from $p_{n,k}$. Provenance is then validated and data is ready for processing. After that, the stored provenance record $p_{n,k}$ of received data packet $d'_{n,k}$ may be deleted from the database, after being used for security analysis. Whereas, if sub-watermarks were not the same, data $d'_{n,k}$ will be discarded and an attack procedure is performed (check the type of attack or origin of data is being altered). The stored provenance $p_{n,k}$ of the discarded data, after attack detection, will be deleted from the database.

Algorithm 2 : Watermark Querying and Verification

```
     input: $d'_{n,k}$
     output: verified/not verified
 1: procedure Watermark querying and verification
 2:     $Receive(d'_{n,k})$
 3:     $R(W'_{n,k}) \leftarrow$ REDO Algorithm 1
 4:     $R(sw'_{h_{n,k}}) \leftarrow EXTRACT(R(W'_{n,k}))$
 5:     $W_{F_{n,k}} \leftarrow QRY(W_{F_{n,k}})$
 6:     $sw_{h_{n,k}} \leftarrow EXTRACT(W_{F_{n,k}})$
 7:     if $(R(sw'_{h_{n,k}}) = sw_{h_{n,k}})$ then
 8:         Data Integrity Verified
 9:         $E(sw_{f_{n,k}}) \leftarrow EXTRACT(W_{F_{n,k}})$
10:         $E(R(sw'_{f_{n,k}})) \leftarrow EXTRACT(R(W'_{n,k}))$
11:         if $(E(R(sw'_{f_{n,k}})) = E(sw_{f_{n,k}}))$ then
12:             Provenance Verified
13:             $p_{n,k} \leftarrow DEC(E(sw_{f_{n,k}}), K_j)$
14:             Extract IoT device $(n)$ IP address
15:             Check provenance information of $(d'_{n,k})$
16:             Process data $(d'_{n,k})$
17:         else
18:             Provenance Not Verified
19:             Discard data $(d'_{n,k})$
20:             Perform attack procedure
21:             Delete $W_{F_{n,k}}$ from network database
22:         end if
23:     else
24:         Integrity Not Verified
25:         Discard data $(d'_{n,k})$
26:         Perform attack procedure
27:         Delete $W_{F_{n,k}}$ from network database
28:     end if
29: end procedure
```

3.4.2. Multi hop scenario

Three algorithms are proposed in this scenario: watermark generation and embedding, watermark verification and re-embedding, and data integrity verification and provenance reconstruction.

1. **Watermark Generation and Embedding:** Algorithm 3 describes the working process of two procedures: watermark generation and storing, and watermark embedding in the multi-hop scenario. The algorithm accepts the captured data $d_{n,k}$ as an input obtained from the source node that is sensing data from the surrounding environment. In the first procedure, two inputs are used for generating the first sub-watermark such as the IoT device IP address w_{ip}, the data sensing time w_t and the generated packet sequence number w_{sq_i} at node n. The sub-watermark $sw_{f_{n,k,i}}$ is formed by appending these values. To secure the provenance information, $sw_{f_{n,k,i}}$ is encrypted using secret key K_j. The encrypted value forms the provenance record $p_{n,k,i}$. Another sub-watermark $sw_{h_{n,k,i}}$ is generated from the hash value of the data payload. Finally, the final watermark $W_{F_{n,k,i}}$ is produced by concatenating the two sub-watermarks $sw_{f_{n,k,i}}$ and $sw_{h_{n,k,i}}$ as in Eq. (2). Provenance record $p_{n,k,i}$ is then stored in the network database as shown in Line 8. In the second procedure, the watermarked data $d_{(n,k)W_{F_{n,k,i}}}$ is produced by concatenating the final watermark $W_{F_{n,k,i}}$ with the captured data packet $d_{n,k}$ as shown in Eq. (3).

$$W_{F_{n,k,i}} = E(w_{ip} \parallel w_t \parallel w_{sq_i}, K_j) \parallel H(d_{n,k}) = E(sw_{f_{n,k,i}}, K_j) \parallel sw_{h_{n,k,i}} \tag{2}$$

$$d_{(n,k)W_{F_{n,k,i}}} = d_{n,k} \parallel W_{F_{n,k,i}} \tag{3}$$

Algorithm 3 : Watermark Generation and Embedding

```
     input: $d_{n,k}$
     output: $d_{(n,k)W_{F_{n,k,i}}}$
 1: procedure Watermark generation and storing
 2:     $w_{ip} \leftarrow$ IoT Device $(n)$ IP Address
 3:     $w_t \leftarrow$ captured data.sensing time $(d_{n,k})$
 4:     $w_{sq_i} \leftarrow seq$ of $(d_{n,k})$ at $n$
 5:     $sw_{f_{n,k,i}} \leftarrow w_{ip} \parallel w_t \parallel w_{sq_i}$
 6:     $p_{n,k,i} \leftarrow E(sw_{f_{n,k,i}}) \leftarrow ENC(sw_{f_{n,k,i}}, K_j)$
 7:     $sw_{h_{n,k,i}} \leftarrow H(d_{n,k})$        ▷ Select first 8 bytes of hash output
 8:     $W_{F_{n,k,i}} \leftarrow p_{n,k,i} \parallel sw_{h_{n,k,i}}$
 9:     $STR(p_{n,k,i})$
10: end procedure
11: procedure Watermark Embedding
12:     $d_{(n,k)W_{F_{n,k,i}}} \leftarrow d_{n,k} \parallel W_{F_{n,k,i}}$
13:     $Send(d_{(n,k)W_{F_{n,k,i}}})$
14: end procedure
```

2. **Watermark verification and re-embedding:** At the next hop, a watermark verification and re-embedding algorithm is applied as shown in Algorithm 4. To verify data integrity at the next node, the algorithm accepts the watermarked data $d'_{(n,k)W_{F_{n,k,i}}}$ as an input. The captured data $d'_{n,k}$ and watermark $W'_{F_{n,k,i}}$ are extracted from $d'_{(n,k)W_{F_{n,k,i}}}$. A new sub-watermark $R(sw_{h_{n,k,i}})$ is re-generated from $d'_{n,k}$ by using the first procedure of Algorithm 3 and $sw'_{h_{n,k,i}}$ is extracted from $W'_{F_{n,k,i}}$. Then a comparison operation is applied on the sub-watermark values of $R(sw_{h_{n,k,i}})$ and $sw'_{h_{n,k,i}}$ to check whether data is altered or not. If data integrity is verified, $E(sw_{f_{n,k,i}})$ is obtained from querying the provenance record $p_{n,k,i}$ from the network database. Another sub-watermark $E(sw'_{f_{n,k,i}})$ is extracted from $W'_{F_{n,k,i}}$ for provenance validation. Then, a comparison operation is applied on $E(sw_{f_{n,k,i}})$ and $E(sw'_{f_{n,k,i}})$. If both sub-watermarks are the same, provenance integrity is verified and a new watermark is generated using the same procedure of Algorithm 3 as shown in Lines 12–19. The new generated watermark $W_{F_{n,k,i}}$ is formed of the next hop node IP address, the watermarked data packet receiving time and a new generated packet sequence number w_{sq_i} of the next hop node, and the same hash value of the data packet obtained from the re-generated sub-watermark $R(sw_{h_{n,k,i}})$ using Eq. (2). The watermark $W_{F_{n,k,i}}$ is concatenated with data $d'_{n,k}$ to form a watermarked data packet as shown using Eq. (3). Then, the new generated sub-watermark $E(sw_{f_{n,k,i}})$ or provenance record $p_{n,k,i}$ is stored in the network database as shown in Line 20. However, if $E(sw_{f_{n,k,i}})$ and $E(sw'_{f_{n,k,i}})$ are not the same, the provenance is not verified and the data is discarded. Also, $P_{n,k}$ of received watermarked data packet $d'_{(n,k)W_{F_{n,k,i}}}$ is deleted from the database and an attack procedure is applied. If data integrity is not verified, data will be also discarded and an attack procedure will be applied. Also, all stored provenance records of $P_{n,k}$ related to this data packet will be deleted from the database.
3. **Integrity verification and provenance reconstruction:** The process of verifying data integrity and reconstructing provenance at the gateway is described in Algorithm 5. The verification procedure relies on five main conditions:

   (a) The origin of data packet based on the source IP address.
   (b) The freshness of the timestamp w_t included in the watermark.

(c) The provenance record sequence number integrity.
(d) The hop by hop integrity and provenance validation.
(e) Verifying the data measurement using the hash value.

The received watermarked data $d''_{(n,k)W_{F_{n,k,i}}}$ is extracted into $d''_{n,k}$ and $W''_{F_{n,k,i}}$. The gateway re-generates the sub-watermark $R(sw_{h_{n,k,i}})$ by performing the generation process of Algorithm 3 as shown in Line 4 Algorithm 5 and $sw''_{h_{n,k,i}}$ is extracted from $W''_{F_{n,k,i}}$. The extracted sub-watermark $sw''_{h_{n,k,i}}$ and the re-generated sub-watermark $R(sw_{h_{n,k,i}})$ will be compared using a comparison operation to check data integrity. If data is not altered, the gateway queries the last provenance record $p_{n,k,i}$ of the received watermarked data packet $d''_{(n,k)W_{F_{n,k,i}}}$ from the database. Then, $E(sw''_{f_{n,k,i}})$ is extracted from $W''_{F_{n,k,i}}$. The gateway performs a comparison operation for $E(sw''_{f_{n,k,i}})$ and $E(sw_{f_{n,k,i}})$ (i.e. last stored provenance record). If both values are the same, provenance is verified, the gateway queries the set of stored provenance records $P_{n,k}$ from the database and extracts the encrypted sub-watermarks $E(sw_{f_{n,k,i}})$ of each $p_{n,k,i}$. At Line 15, the secret key K_j is used to decrypt $E(sw_{f_{n,k,i}})$ and obtain the sub-watermarks $sw_{f_{n,k,i}}$ containing provenance information of the received data packet. The gateway constructs the data path from provenance information obtained and uses packet sequence record to identify any provenance record drop attack or any modification in the packet forwarding path. If data integrity or provenance is not verified, data will be discarded and an attack procedure is performed and $P_{n,k}$ of received watermarked data packet $d''_{(n,k)W_{F_{n,k,i}}}$ is deleted from the database.

Algorithm 4 : Watermark Verification and Re-embedding

```
    input: $d'_{(n,k)W_{F_{n,k,i}}}$
    output: verified/not verified, $W_{F_{n,k,i}}$, $d'_{(n,k)W_{F_{n,k,i}}}$
 1: procedure Watermark verification and re-embedding
 2:     $Receive(d'_{(n,k)W_{F_{n,k,i}}})$
 3:     Extract Watermarked Data into $d'_{n,k}$ and $W'_{F_{n,k,i}}$
 4:     $R(sw_{h_{n,k,i}}) \leftarrow$ REDO Algorithm 1
 5:     $sw'_{h_{n,k,i}} \leftarrow EXTRACT(W'_{F_{n,k,i}})$
 6:     if $(R(sw_{h_{n,k,i}}) = sw'_{h_{n,k,i}})$ then
 7:         Integrity Verified
 8:         $E(sw_{f_{n,k,i}}) \leftarrow QRY(p_{n,k,i})$
 9:         $E(sw'_{f_{n,k,i}}) \leftarrow EXTRACT(W'_{F_{n,k,i}})$
10:         if $(E(sw'_{f_{n,k,i}}) = E(sw_{f_{n,k,i}}))$ then
11:             Provenance Integrity Verified
12:             Generate next hop watermark{
13:                 $w_{ip} \leftarrow$ next hop device IP Address
14:                 $w_t \leftarrow$ received time $(d'_{(n,k)W_{F_{n,k,i}}})$
15:                 $w_{sq_i} \leftarrow (seq(d'_{(n,k)}))$          ▷ new sequence number
16:                 $i++$                    ▷ update next hop watermark index
17:                 $sw_{f_{n,k,i}} \leftarrow w_{ip} \,||\, w_t \,||\, w_{sq_i}$
18:                 $p_{n,k,i} \leftarrow E(sw_{f_{n,k,i}}) \leftarrow ENC(sw_{f_{n,k,i}}, K_j)$
19:                 $sw_{h_{n,k,i}} \leftarrow$ Hash value from $R(sw_{h_{n,k,i}})$
20:                 $W_{F_{n,k,i}} \leftarrow E(sw_{f_{n,k,i}}) \,||\, sw_{h_{n,k,i}}$}
21:             $d'_{(n,k)W_{F_{n,k,i}}} \leftarrow d'_{n,k} \,||\, W_{F_{n,k,i}}$
22:             $STR(p_{n,k,i})$
23:             $Send(d'_{(n,k)W_{F_{n,k,i}}})$
24:         else
25:             Provenance no verified/attack detection
26:             Discard data $d'_{n,k}$ & Perform attack procedure
27:             Delete $P_{n,k}$ from network database
28:         end if
29:     else
30:         Not verified/attack detection
31:         Discard data $d'_{n,k}$
32:         Perform attack procedure
33:         Delete $P_{n,k}$ from network database
34:     end if
35: end procedure
```

Algorithm 5 : Watermark Restoring and Verification

```
    input: $d''_{(n,k)W_{F_{n,k,i}}}$
    output: verified/not verified, provenance reconstruction
 1: procedure Watermark Restoring and Verification
 2:     $Receive(d''_{(n,k)W_{F_{n,k,i}}})$
 3:     Extract Watermarked Data into $d''_{n,k}$ and $W''_{F_{n,k,i}}$
 4:     $R(sw_{h_{n,k,i}}) \leftarrow$ REDO Algorithm 3
 5:     $sw''_{h_{n,k,i}} \leftarrow EXTRACT(W''_{F_{n,k,i}})$
 6:     if $(R(sw_{h_{n,k,i}}) = sw''_{h_{n,k,i}})$ then
 7:         Data integrity verified
 8:         $E(sw_{f_{n,k,i}}) \leftarrow QRY(p_{n,k,i})$
 9:         $E(sw''_{f_{n,k,i}}) \leftarrow EXTRACT(W''_{F_{n,k,i}})$
10:         if $(E(sw_{f_{n,k,i}}) = E(sw''_{f_{n,k,i}}))$ then
11:             Provenance integrity verified
12:             $P_{n,k} \leftarrow QRY(P_{n,k})$
13:             for (index $i$, $1 \le i \le H$, $i++$) do
14:                 Extract $E(sw_{f_{n,k,i}})$ of each $p_{n,k,i}$ from $P_{n,k}$
15:                 $sw_{f_{n,k,i}} = DEC(E(sw_{f_{n,k,i}}), K_j)$
16:                 Extract provenance information
17:             end for
18:             Construct data path of $d''_{n,k}$
19:         else
20:             Provenance integrity is not verified
21:             Attack detection
22:             Discard data $d''_{n,k}$
23:             Perform attack procedure
24:             Delete $P_{n,k}$ from network database
25:         end if
26:     else
27:         Data integrity not verified/ attack detected
28:         Discard data $d''_{n,k}$
29:         Perform attack procedure
30:         Delete $P_{n,k}$ from network database
31:     end if
32: end procedure
```

3.5. Managing internal datagrams

In this section, we propose the idea of labeling IP datagrams that are used internally for network management. These datagrams should not be analyzed by the IDS and will undergo an internal security procedure. This optimizes the scheme by minimizing the number of IDS operations on data packets. The advantage of this protocol is the use of the Identification field, flags and fragment offset as the embedding positions in the IP datagram header which will appear random-like and will not show an evident pattern that an attacker may try to exploit (cf. Section 3.3). The management of IP datagrams by network nodes is formally described in Algorithms 6 and 7.

At each node or gateway, internal managing packets are labeled with a hash value that is computed and embedded before the packet is sent. The hash value is computed as H(Destination IP || First 20 bytes

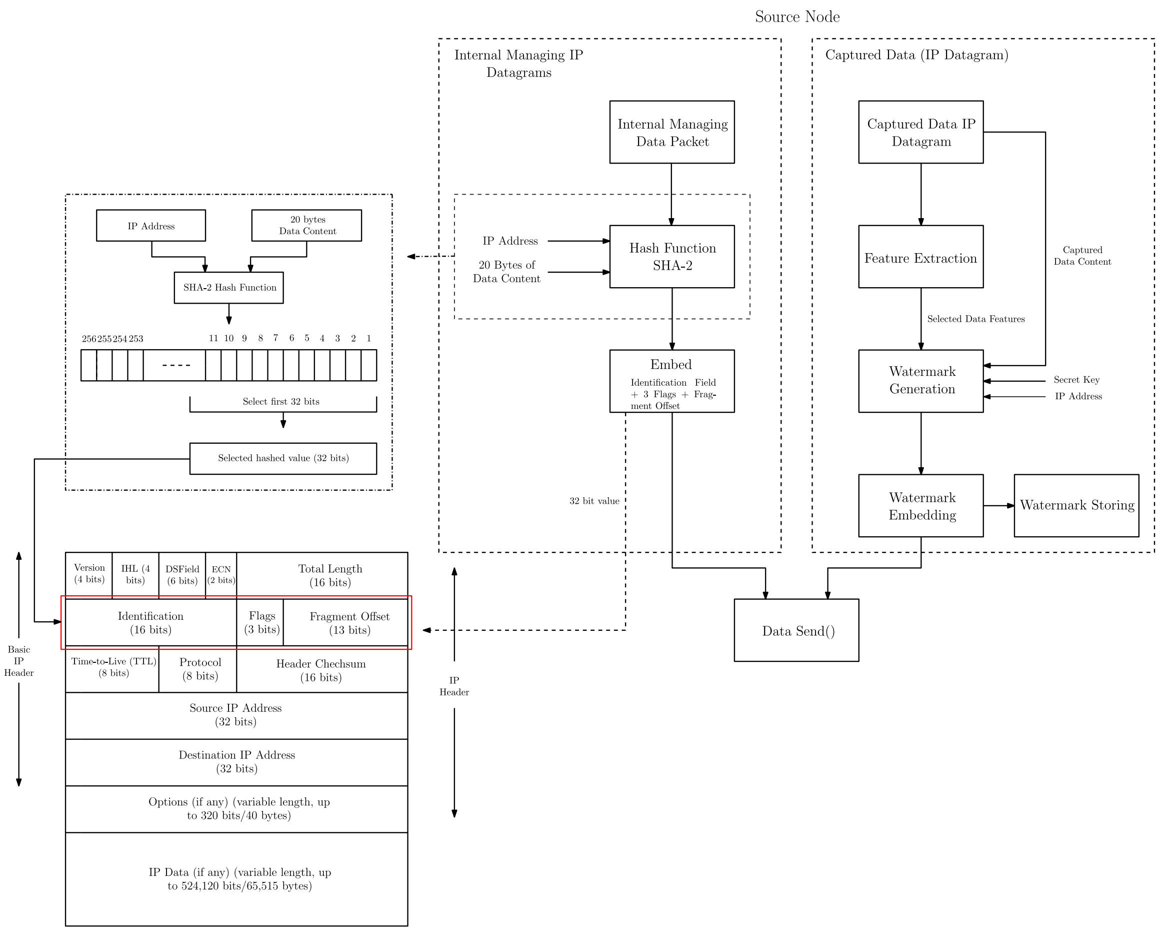


Fig. 6. Embedding hash value in the Internal Managing Protocol. The red square represents the embedding position of the selected bit from the generated hash function. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 6 : Internal Managing at the Source Node

```
    input: IP datagram $d_{IP}$
    output: Embedding hash value
1: procedure INTERNAL MANAGING EMBEDDING PROCESS
2:     if($d_{IP}$ = internal managing packet) then
3:         Compute H(Destination IP || First 20 bytes of $d_{IP}$)
4:         $d_{IP}$(header) ← H(Dest. IP || First 20 bytes of $d_{IP}$)
5:     else
6:         perform watermark generation and embedding
7:     end if
8: end procedure
```

Algorithm 7 : Internal Managing at the Destination Node

```
    input: IP datagram $d_{IP}$
    output: Require IDS/internal-managing
1: procedure INTERNAL MANAGING PROCESS
2:     Receive ($d_{IP}$)
3:     if(IP datagram ← ($src$, $dest$)) then
4:         if(($src$, $dest$ = internal) & L($R_D$) = L($D$)) then
5:             Compute H(Destination IP || First 20 bytes of $d_{IP}$)
6:             Extracted H ← $d_{IP}$(Identification+Flags+Offset)
7:             if(Computed H = Extracted H) then
8:                 $d_{IP}$ is authenticated, i.e.,
9:                 $d_{IP}$ is not examined by the IDS
10:            else
11:                attack detection
12:                $d_{IP}$ is discarded
13:            end if
14:        else
15:            $d_{IP}$ must be examined by the IDS
16:        end if
17:    else
18:        $d_{IP}$ must be examined by the IDS
19:    end if
20: end procedure
```

of the datagram content), where the operator || denotes concatenation. The value is then embedded in the IP datagram header as shown in Algorithm 6 and Fig. 6. We use the Identification field (16 bits), Flags (3 bits) and Fragment offset (13 bits) to embed the selected 32 bit from the hash value. After receiving any IP Datagram at the Gateway or any node in the internal network, an internal managing protocol is performed (before any IDS procedure) as shown in Algorithm 7. The datagram is subjected to a first condition that checks whether these datagrams have both a source and a destination address in our local network, since this is a first condition (filter). Then it checks if both the source and the destination address are internal and the size of the received data packet is equal to an internal managing packet size. Sensed data packets by sensor nodes are watermarked and have different size (data packet + watermark) as shown using Eq. (3). If it is not the case, the datagram must be examined by the IDS (it is not an internal data packet). However, if both IP addresses are internal and the size is confirmed, the device computes H(Destination IP address || First 20 bytes of the datagram content) and extracts the hash value embedded in the header of the data packet. Then the node compares these two hash values. If these values are the same, the datagram is authenticated as "authorized internal-managing packet". Otherwise, an attack is detected and the datagram is discarded.

The hash function used in obtaining the IP datagram label is SHA-2. SHA-2 takes an input of any size and produces a 256-bit hash value. Since the Identification field is 16 bit long and the size of Flags and offset take another 16 bits, making a total of 32 bits, the device selects 32 Least Significant Bit (LSB) bits from the hash value as shown in Fig. 6. We can also randomize the selection of these 32 bits by using pseudo-random number generator and obtain randomized bit positions that can be selected. This randomization would add another level of security for the system.

4. Security analysis

The IoT network can be subject to two main security breaches in the transmission phase: passive and active attacks on both data and watermark. An adversary can launch various attacks based on the threat model described in Section 3.3. In this section, we provide an analysis for the security of the proposed scheme against the attacks detailed in Section 3.3. We assume that the network gateway and database are trusted and cannot be compromised by an attacker.

Claim 1. *An unauthorized party cannot access or obtain the secret information generated by the source node S_n.*

Rationale. The source node S_n generates a final watermark $W_{F_{n,k,i}}$ by concatenating two sub-watermarks $sw_{f_{n,k,i}}$ and $sw_{h_{n,k,i}}$. The first sub-watermark $sw_{f_{n,k,i}}$ is obtained from extracted data features as follows: IP address w_{ip}, sensed data capturing time w_t and data packet sequence number w_{sq}. These data features are encrypted using Advanced Encryption Standard (AES) algorithm using a symmetric secret key K_j. $sw_{f_{n,k,i}}$ is generated and encrypted as $E(sw_{f_{n,k,i}}) = ENC(sw_{f_{n,k,i}}, K_j)$. Thus, an attacker being unaware of K_j cannot decrypt $sw_{f_{n,k,i}}$ (only authorized parties are aware of K_j, i. e., intermediate nodes and gateway). Note that K_j is changed and redistributed after a short random number of watermark generation sessions (see Section 3.3). For the second sub-watermark $sw_{h_{n,k,i}}$, a source node uses a one-way cryptographic hash function $H()$ to obtain $sw_{h_{n,k,i}}$ used for data integrity check. It is computationally infeasible to find a pair (x, y) such that $h(x) = h(y)$, which make the function secure and cannot be inverted as assumed in Section 3.3. Additionally, we use SHA-2 hash function in our scheme with 256 bit hash value, which make it computationally infeasible for an attacker to carry out 2^{128} calculations to find the second sub-watermark. The generated watermark is computed as $W_{F_{n,k,i}} = E(sw_{f_{n,k,i}}) \parallel sw_{h_{n,k,i}}$ for each captured data. Thus, an adversary cannot access watermark information generated by source nodes.

Claim 2. *An attacker, cannot successfully deceive an intermediate node I_l or gateway G by inserting fake data or deleting data from the data flow generated by a legitimate node S_n and transmitted to I_l or G.*

Rationale. In case an attacker inserts fake data into a watermarked data-packet $d_{(n,k)W_{F_{n,k,i}}}$ being transmitted to I_l or G, the destination node extracts $d_{(n,k)W_{F_{n,k,i}}}$ into sensed data $d_{n,k}$ and watermark $W_{F_{n,k,i}}$. Then, a re-generated sub-watermark $R(sw_{h_{n,k,i}})$ is computed from the received captured data $d_{n,k}$ and compared to the extracted watermark $sw'_{h_{n,k,i}}$ from $W_{F_{n,k,i}}$. The process of re-generation is based on the previously mentioned generation process (i. e., SHA-2 hash function for $sw_{h_{n,k,i}}$). Hence, any change in $d_{n,k}$ content produces an altered re-generated sub-watermark. The assumption of secure communication of extracted data features and provenance information using symmetric cryptography and a one-way hash function applies (Section 3.3). Then, even if an attacker inserts fake data into $W_{F_{n,k,i}}$ without altering $d_{n,k}$, $R(sw_{h_{n,k,i}})$ will not match $sw'_{h_{n,k,i}}$ in the comparison process. Also, if the attacker inserts fake data to the second sub-watermark $E(sw'_{f_{n,k,i}})$ the next hop intermediate node or gateway queries the stored provenance record $p_{n,k,i} = E(sw_{f_{n,k,i}})$ from the data base and compares it with the extracted sub-watermark $E(sw'_{f_{n,k,i}})$. Any change in $E(sw'_{f_{n,k,i}})$ yields to alternation in the provenance information. In the second case, the attacker aims to delete data content from $d_{n,k}$ or $W_{F_{n,k,i}}$, or drop an entire data-packet $d_{(n,k)W_{F_{n,k,i}}}$ being routed from S_n to I_l or from I_l to G. The deletion of q bits from $d_{n,k}$ results in the modification of $R(sw_{h_{n,k,i}})$ and thus $sw'_{h_{n,k,i}}$ will not match $R(sw_{h_{n,k,i}})$. Again, the previously mentioned assumption of secure communication of $W_{F_{n,k,i}}$ applies. Furthermore, if the attacker deletes q bits from $W_{F_{n,k,i}}$ it will be detected in the comparison process of the two sub-watermarks $R(sw_{h_{n,k,i}})$ and $sw'_{h_{n,k,i}}$ or between the queried sub-watermark $E(sw_{f_{n,k,i}})$ and the extracted one $E(sw'_{f_{n,k,i}})$. Obviously, such an adversary may drop $d_{(n,k)W_{F_{n,k,i}}}$ routed through I_l. This attack can be detected at G by accessing the tamper-proof database and querying the provenance records of $d_{n,k}$, and detecting where the packet drop attack occurred. The database stores provenance records securely, which cannot be accessed by an attacker (as described in Section 3.3).

Claim 3. *An attacker, attempting to alter provenance information: (i) cannot add legitimate nodes to the provenance of data generated by an unauthorized node, (ii) cannot successfully add or remove nodes from the provenance of data generated by legitimate nodes.*

Rationale. I_l stores a provenance record $W_{F_{n,k,i}}$ after checking data integrity and provenance of the received data-packet $d_{(n,k)W_{F_{n,k,i}}}$. The symmetric secret key K_j shared between legitimate nodes is used to obtain the generated watermark $W_{F_{n,k,i}}$ used in data integrity and provenance validation. An unauthorized node generates watermarks using its own secret key that cannot match a generated watermark at I_l using K_j. As stated in Section 3.3, the source node, the intermediate node and the gateway share secret-keys to be used in different steps of the algorithms (encryption/decryption). These keys are changed and redistributed between legitimate nodes after a random number of sessions. Thus, in order to add a legitimate node, an attacker needs to obtain the same symmetric secret key that is only shared within legitimate network nodes of internal registered IP addresses. In the case of two malicious nodes I_m and I_v attempting to execute an attack, a captured data-packet $d_{n,k}$ by a legitimate source node S_n is routed through S_m. $d_{n,k}$ has a provenance record of (I_1, I_2, I_4). The malicious node I_m aims to remove I_2 and replace it with I_v. To add I_v as a provenance record to the database, the malicious node needs to compute the next-hop watermark which requires, as mentioned above, the knowledge of K_j and hash function variables. Hence, the provenance integrity check at the next I_j will fail and an attack is detected. Thus, I_m will fail to add or remove any provenance record from network database. Moreover, provenance records $(W_{F_{n,k,1}}, W_{F_{n,k,2}}, \ldots, W_{F_{n,k,i}})$ of a data-packet $d_{n,k}$ are stored in a tamper-proof database that is assumed to be resistant to any alternation of its entities, attackers cannot alter any record stored in it (see Section 3.3).

Claim 4. *It is impossible for an attacker, whether acting alone or in collaboration with others, to add or authenticate nodes to the provenance of data produced by a compromised node.*

Rationale. An attacker may generate fake data and store provenance information in the database as a legitimate node with its secret key. The packet is then forwarded to the next hop intermediate node to store the next hop provenance information in the set of provenance records $P_{n,k}$ for this data packet in the database. The attacker's aim is to construct the provenance from innocent forwarding nodes and make them responsible for false data forwarding, thus marking them as untrustworthy nodes. However, there is an integrity and provenance validation procedure at the next hop node, which includes a watermark re-generation process $W_{F_{(n,k,i)}}$ using the secret key K_j, the attacker do not know the key for legitimate nodes. Thus, this attack will fail at the first hop.

Claim 5. *Any unauthorized attempt to modify data content through transmission channel would be detected.*

Rationale. An adversary may perform a modification to the embedded watermark (computed as $W_{F_{n,k,i}} = E(sw_{f_{n,k,i}}) \parallel sw_{h_{n,k,i}}$) or data elements $d_{(n,k)W_{F_{n,k,i}}}$. If data elements are modified and $W_{F_{n,k,i}}$ remains unchanged, a different watermark is obtained based on a wrong hash value at an intermediate node or gateway. Since the first generated sub-watermark at source node $sw_{h_{n,k,i}}$ is the output of a hash function SHA-2 obtained as $sw_{h_{n,k,i}} = H(d_{n,k})$. Again, the assumption of hash functions used in the system (Section 3.3) applies. The wrong re-generated sub-watermark $R(sw_{h_{n,k,i}})$ will not match the extracted sub-watermark $sw'_{h_{n,k,i}}$. The intermediate node or gateway detects the modification attack and discards the data. Furthermore, if the attacker modifies $W_{F_{n,k,i}}$ and the data payload remains unchanged, the intermediate node or gateway re-generates the right sub-watermark, extract the modified watermark from the received data packet and queries the provenance record $p_{n,k,i}$ from the database. This results in a failed comparison operation for data integrity or for provenance validation and data will be discarded.

Claim 6. *By including a timestamp in the generation process of watermarks, any fraud transmission of previously captured data packets will be discovered.*

Rationale. An attacker may provide a false idea about the sensing environment by fraudulently transmitting previously heard data packets that are captured and transmitted by a legitimate source node [43]. The attacker also detects the timing characteristics to be used later during the packet replay attack. To deceive an intermediate node or gateway, the attacker updates the timestamp w_t of the heard data packet $d_{n,k}$, based on timing characteristics, to a new recent time value. In the proposed scheme, a source node generates a watermark $W_{F_{n,k,i}}$ for each data packet captured ($d_{n,k}$). The generation process is based on provenance information, a timestamp and a hash value as described in Eq. (1). Provenance information and timestamp will be encrypted using a secret key K_j to form the first sub-watermark (i. e., encrypting $sw_{f_{n,k,i}} = w_{ip} \parallel w_t$ as $E(sw_{f_{n,k,i}}) = ENC(sw_{f_{n,k,i}}, K_j)$). At next hop I_l or G, a new sub-watermark is generated from the replayed packet that will be compared to the extracted sub-watermark. If the attacker changed the timestamp of the data packet $d_{n,k}$ the comparison operation will fail. Since timestamps are different the new re-generated sub-watermark will not match the extracted one. Note that the attacker cannot modify the timestamp w_t embedded in the watermark $W_{F_{n,k,i}}$, due to the encryption process performed on the generated sub-watermark $sw_{f_{n,k,i}}$. The sub-watermark is encrypted using the source secret key K_j, which is only shared with legitimate entities (intermediate node and gateway) where an attacker uses a different secret key as stated in Section 3.3. Hence, replaying an old packet with an updated timestamp will lead to a failed authentication procedure.

Claim 7. *Any attempt from an attacker to selectively drop a provenance record from the database or alter the provenance will be detected at the base station.*

Rationale. If an attacker manages to compromise the database and selectively remove a provenance record $p_{n,k,i}$ from a data packet's provenance $P_{n,k}$, the base station will query the complete provenance information from the database. This query occurs after a final integrity validation of both the data payload and provenance. After decrypting the retrieved sub-watermarks of $P_{n,k}$, the base station extracts the provenance information and checks the IP address w_{ip}, timestamp w_t, and packet sequence number w_{sq_i} of each provenance record. This information is used to construct the data path. If any provenance records are missing or have out-of-order sequence numbers in the stored forwarding provenance records, they will be detected. Consequently, the base station is able to identify any provenance record dropping attack. This method is also applicable in the single-hop scenario, where the base station detects packet drop attacks using the sequence number w_{sq} stored in the provenance record of each data packet in the network database as shown in Algorithm 1.

Claim 8. *In Algorithm 7, an attacker trying to deceive network devices to accept malicious datagrams as trusted internal managing datagrams will be detected and examined by implemented security algorithms.*

Rationale. If an attacker succeeds to modify an internal managing datagram, the datagram will be forwarded to the next hop node. At the receiving node, a hash value is computed from the content of the datagram using a one way hash function as detailed in Section 3.5. It also extracts the hash value embedded by the source node from the identification field of the IP header. Both hash values are then compared to detect any attempt of forgery attack. If the values do not match, the device applies the implemented security algorithms to the received IP datagram and an attack procedure is performed. Note that a source node uses a one-way cryptographic hash function $H()$ using SHA-2 to obtain the hash value (embedded in internal managing IP datagram's header) so that it is computationally infeasible to find a pair (x, y) such that $h(x) = h(y)$, making it impossible for an attacker to invert the hash value and embed it to deceive the system (Section 3.3). Hence, a malicious entity trying to deceive the forwarding nodes using internal managing datagrams will be detected and discarded.

Claim 9. *The proposed scheme demonstrates robustness against DoS attacks.*

Rationale. A DoS attack on an IoT network is a malicious attempt to disrupt the normal functioning of the network by overwhelming it with a flood of illegitimate requests or traffic. This type of attack can render IoT devices or services unavailable to legitimate users by exhausting the network's resources, such as bandwidth, processing power, or memory. The characteristics and impact of DoS attacks on IoT networks are as follows:

- *Resource Limitation:* IoT devices typically have limited computational resources, memory, and bandwidth. This makes them particularly vulnerable to DoS attacks as they can be easily overwhelmed by a relatively low volume of malicious traffic compared to traditional network devices.
- *Diverse and Distributed Nature:* IoT networks often consist of a vast number of heterogeneous devices distributed across various locations, making it challenging to secure the entire network effectively and to identify and mitigate attacks promptly.
- *Critical Applications:* Many IoT applications, such as smart grids, healthcare monitoring systems, and industrial control systems, are critical and require high availability and reliability. A DoS attack on such networks can lead to significant disruptions

DoS attacks on IoT networks can be launched using various methods:

1. *Flooding Attacks:* Attackers send an overwhelming amount of traffic to the target device or network, consuming its bandwidth and processing capacity. Flooding attacks include:

    - *HTTP Flooding:* Overloading the device with HTTP requests.
    - *UDP Flooding:* Sending a large number of User Datagram Protocol (UDP) packets.
    - *TCP SYN Flooding:* Exploiting the TCP handshake process by sending numerous SYN requests without completing the handshake.

2. *Exploitation of Vulnerabilities:* Attackers exploit specific vulnerabilities in the IoT devices' firmware or software to cause them to crash or become unresponsive. This exploitation includes:

    - *Buffer Overflow:* Sending specially crafted packets that overflow the buffer memory of the device, causing it to crash.
    - *Firmware Exploits:* Targeting known vulnerabilities in the device firmware to disrupt its operation.

ZIRCON can mitigate DoS attacks on IoT networks by ensuring the authenticity and integrity of the data packets at each hop and gateway. By embedding the watermark $W_{F_{n,k,i}}$, the system can verify the source and legitimacy of each packet, discarding any that fails the verification process as described in Algorithms 4 and 5. This prevents malicious packets from overwhelming the network, as only authenticated traffic is allowed to pass through. The continuous verification at each hop and the gateway helps in early detection and filtering of illegitimate traffic, thus protecting the network from being flooded with malicious data such as HTTP, UDP and TCP SYN flooding attacks. Since the approach verifies the authenticity of the source at each hop, it prevents attackers from easily injecting illegitimate traffic into the network. Attackers would need access to encryption keys and provenance information used in watermark generation and verification process, which is significantly more challenging. Each intermediate node I_l can verify if the packet has traversed legitimate source nodes S_n, ensuring that the packet's data path or provenance $P_{n,k}$ through the network is as expected and not deviated from normal behavior. Any deviation can trigger an alert or the dropping of the packet d_n, preventing malformed or spoofed packets from consuming network resources. Hence, ZIRCON verification process at each hop and at the gateway filters out packets without valid watermarks, thereby reducing the bandwidth and processing load on IoT devices and preventing them from being overwhelmed by illegitimate traffic.

Claim 10. *An attacker aiming at the interception of data communication between IoT devices using MITM attacks can be detected.*

Rationale. Many IoT devices have weak security features and lack the processing power to implement complex encryption protocols. A MITM attack is a serious threat to these devices on an IoT network. In this attack, an attacker secretly inserts themselves into the communication between two devices. This allows them to eavesdrop on the conversation, steal sensitive data, or even modify the data being exchanged. The attacker intercepts communication between the IoT devices and the legitimate destination (maybe I_l or G). This can be done through various methods like ARP spoofing or setting up a fake WiFi network. Once in the middle, the attacker can listen to the data flowing between the devices. This could include sensitive information such as sensor data or control commands (internal packets). The attacker can also modify the data before it reaches its destination, potentially causing malfunctions or disrupting operations. Also, the attacker can modify the intercepted packets to inject false data, alter commands, or introduce malicious payloads into the communication data stream. In our scheme, by embedding watermarks into data packets using provenance information, ZIRCON ensures that any alteration of the watermarked data packets $d_{(n,k)W_{F_{n,k,i}}}$ will either modify the watermark $W_{F_{n,k,i}}$ or alter the packet payload. At the next hop or gateway, the data packet undergoes verification. If a packet's watermark is invalid or altered, it is identified as compromised and the packet is discarded. Hence, the process of verifying the watermark at each hop means that even if an attacker intercepts and modifies a packet between two nodes, the modification will be detected at the next node, preventing the altered packet from proceeding further. Moreover, provenance information helps ensure that the packet is from a legitimate source node S_n. A MITM attacker cannot easily forge provenance information because it is cryptographically bound to the packet by a secret key K_j. The use of encryption keys to embed watermarks means that an attacker would need access to these keys to generate or modify a valid watermarks. Without the key, any attempt to alter the packet will result in an invalid watermarks, making it easier to detect and discard tampered packets.

Based on the above analysis, the proposed scheme is proven to be resistant against various malicious attacks of IoT networks, such as modification attack, integrity attack, packet replay, database authentication attack, DoS, MITM and passive attacks. It guarantees the integrity of data and ensures security against identifying and retrieving provenance information in IoT networks.

5. Simulation results and analysis

In this section, we evaluate the performance of the proposed scheme based on two features: data integrity and data provenance. For data integrity, the proposed scheme is evaluated based on watermark generation, embedding and verification time. Also, we have measured how this scheme performs in terms of energy usage. The results are then compared to three state of the art techniques: RWFS [32], Asymmetric Cryptography Technique (ACT) [30] and Zero-Watermarking Scheme (ZWT) [35] based methods. We selected these three state-of-the-art methods to assess the performance of our new security technique based on their use of different security techniques deployed in a similar network model. For data provenance, we compare our scheme with MAC-based provenance scheme (MP), a secure provenance framework $SProv$ [44], and a lightweight secure scheme BFP [20] in terms of cost analysis. The algorithms were implemented in MATLAB™ on Intel core i7 processor with a 2.59 GHz clock cycle and 16 GB of memory. Sensor data is represented as an integer data type, since most sensor readings are of numeric form such as temperature, humidity, motion and intensity.

In our algorithm, we use AES with 128 bit key size for encryption of generated watermarks. Despite the fact that AES has a larger key size than Data Encryption Standard (DES), AES is a more secure and advanced encryption algorithm compared to DES, which makes it more resistant to cryptanalysis attacks. Another reason for using AES is its performance and efficiency. AES is a fast and efficient algorithm. We provide, in Figs. 7(a), 7(b), 8(a) and 8(b), a comparison of using AES and DES algorithms in the generation and verification processes at each sensor node in the proposed model. The results show the better performance of our scheme when applying AES algorithm (approximately 10 times faster) in both watermark generation and verification. The use of substitution-permutation network (SPN) structure, which is optimized for hardware implementation and allows for parallel processing in AES shows a faster performance than DES, which uses a Feistel network structure. Regarding the hash function, we use a one way hash function SHA-2, specifically SHA-256, for generating the second sub-watermark sw_h. Although SHA-1 is faster than SHA-2 functions since it uses a smaller block size and has a simpler construction, however it is important to note that the slower performance of SHA-2 functions is outweighed by their improved security compared to SHA-1. We compared the generation and verification time of the proposed model using different hash functions in Figs. 9(a) and 9(b). The results shows

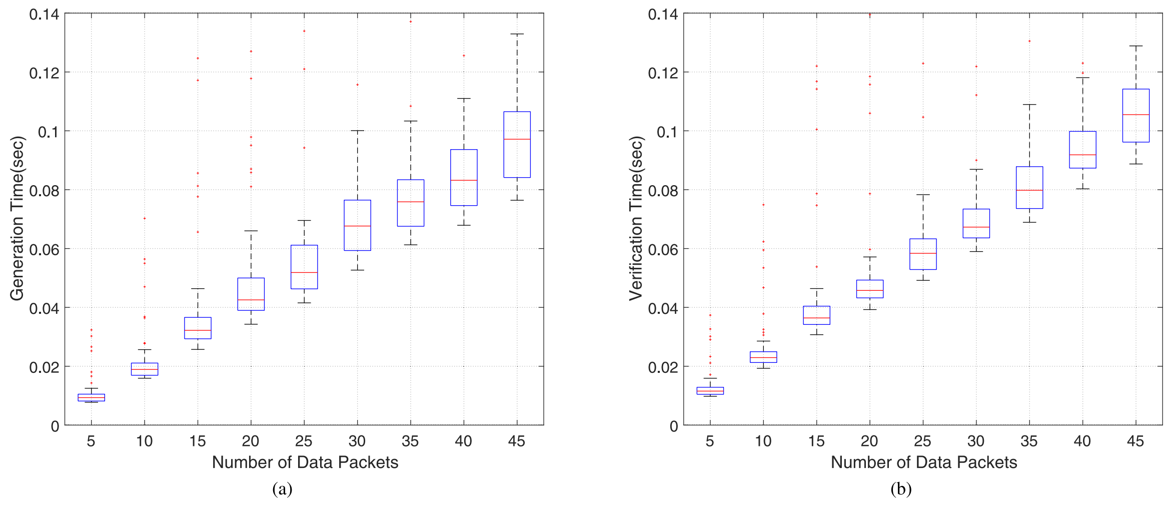


Fig. 7. Computational time. (a) Watermark generation and embedding time using AES. (b) Watermark verification time using AES.

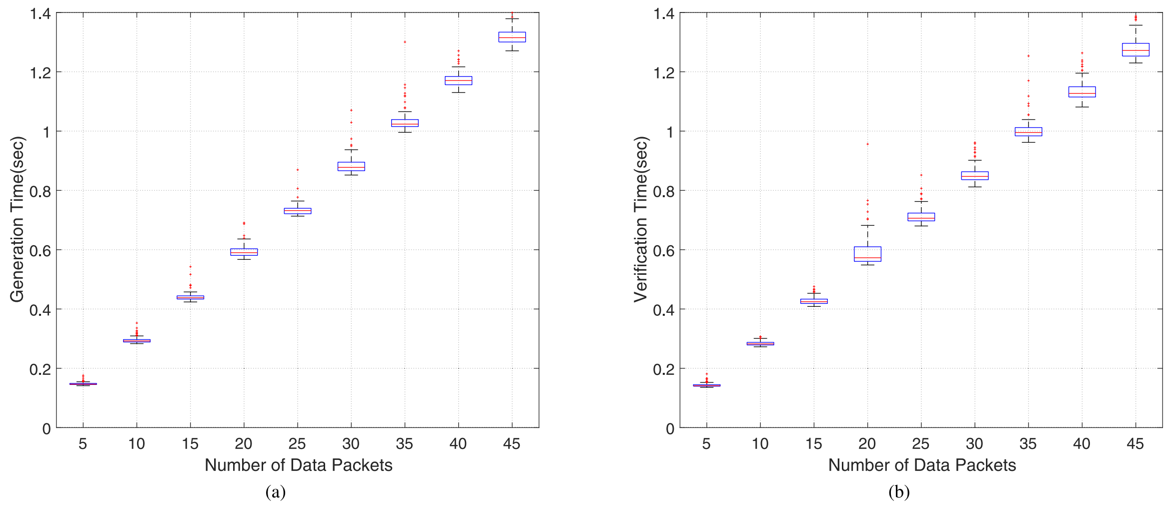


Fig. 8. Computational time. (a) Watermark generation and embedding time using DES. (b) Watermark verification time using DES.

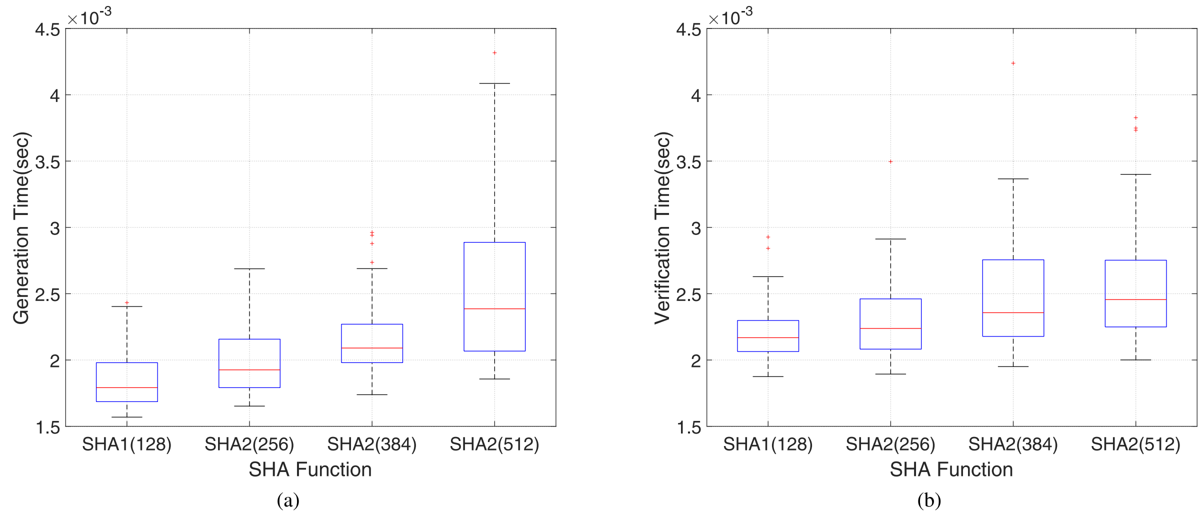


Fig. 9. SHA Comparison. (a) Watermark generation and embedding time using different SHA functions. (b) Watermark verification time using different SHA functions.

that SHA-1 is faster than SHA-2 functions and SHA-2(256) function requires less processing time than SHA-2(384) and SHA-2(512). Hence, we use SHA-2(256), which provides the best performance in SHA-2 functions, as our hash function in the generation of watermarks.

5.1. *Performance evaluation*

To evaluate the performance, we measure the computational time such as watermark generation, embedding, and watermark verification time of the proposed scheme, RWFS [32], ACT [30] and ZWT [35]. Additionally, we used energy consumption as another performance metrics and compared the results with existing methods [30,32,35]. We select these three works from the literature to compare our model with a regular watermarking technique, asymmetric cryptography technique and a zero-watermarking technique. From our research work, these papers provide these three methods and deploys it in a scenario similar to what we are analyzing and studying. Note that a confidence interval is added to show the average generation and verification time after a 100 simulation runs.

5.1.1. *Computational time*

Computational time is described as the time required to complete the following processes: watermark generation, embedding and verification at sensor nodes and gateway.

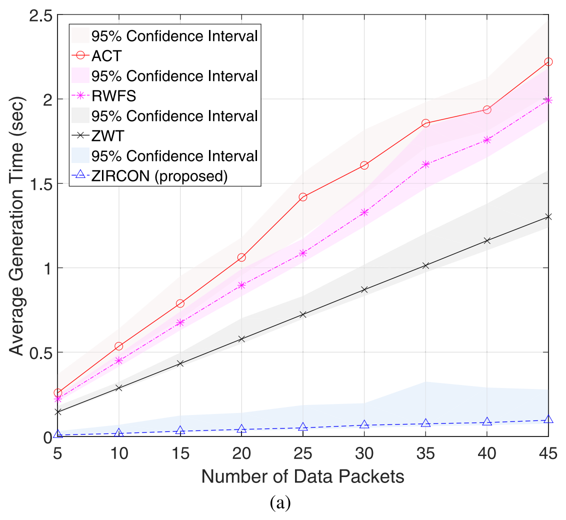
1. **Watermark generation and embedding time:** This metric measures the time taken by a node to generate a watermark and embed it into the data packet. It reflects the efficiency of the watermark generation and embedding process, which is important for real-time applications where delays must be minimized. Faster watermark generation and embedding reduce the end-to-end processing time, improving the responsiveness of the system. This is particularly important in applications like IoT, where timely data processing is essential. The existing RWFS [32] generates a watermark by encrypting the sensed data with a homomorphic encryption algorithm proposed by Castelluccia et al. [45] and passing it as an input to a keyed-hash message authentication code (HMAC). The watermark is then embedded randomly by computing each position of watermark bits using a pseudo-random number generator (PRNG) for each captured data at source node. In ACT [30], the watermark generation is based on an asymmetric cryptography function and uses group hashing for a set of data values that need to be captured in different time intervals before generating the watermark. Additionally, ZWT [35] uses DES for watermark encryption in the watermark generation process. Comparing these approaches [30,32,35], the proposed scheme uses a zero-watermarking technique that generates a fixed size watermark from provenance information and data features. It applies a one-way hash function to extracted data features and symmetric encryption (i. e., AES) for provenance information. Simulation results shows that the proposed scheme requires less watermark generation and embedding time than existing approaches [30, 32,35] as observed in Fig. 10(a). Using AES as an encryption and SHA-2 to generate watermarks shows a significant improvement in the performance of sensor nodes. This results in decreasing the end-to-end time from capturing data to processing it at the destination gateway.
2. **Watermark verification time:** The verification algorithm is used to extract watermark and verify data integrity at the destination node. This procedure is performed at an intermediate node or gateway which have more computational and power capabilities than source nodes. Hence, this metric evaluates the time needed to extract and verify the watermark at the destination node. It indicates the efficiency of the verification process in ensuring data integrity. Shorter verification times enhance the overall system performance by reducing the processing overhead at destination nodes. They also contribute to lower latency in data verification, which is critical for time-sensitive applications. In the proposed approach, the watermark is concatenated to the data payload and each watermark is generated using AES and SHA-2 for each data packet which requires less extraction and verification time than RWFS [32], ACT [30] and ZWT [35]. The time for extracting and verifying data integrity in [32] depends on computing each watermark bit position and computing a hash value after re-encrypting the extracted data. In [30], the intermediate node or gateway requires receiving several data packets to perform watermark extraction and re-calculating the watermark based on asymmetric encryption to perform verification. Moreover, in ZWT [35], the intermediate node or gateway needs to extract data features from the received data packet and encrypt these features using DES algorithm to re-generate the watermark for verification. Fig. 10(b) shows that the proposed zero-watermark approach requires less time to extract and verify data integrity than existing schemes [30,32,35]. It is worth pointing out that the proposed approach provides both data integrity and data provenance. The time shown in Fig. 10(b) for the proposed scheme includes also the time needed for querying the stored watermarks from the database.

5.1.2. *Energy consumption*

Energy consumption evaluates the energy consumed by a sensor node from the power utilized by each node and the total time consumed in the sensor node operation steps as shown in Fig. 13. The energy consumed by a sensor node varies based on several basic energy consumption sources: processing time cost, radio transmission, sensor sensing, transient energy, and sleeping time cost [46–48]. It is crucial to utilize less energy-consuming security mechanisms for IoT networks due to the limited computation and power capabilities of sensor nodes. Energy consumption plays a critical role in the viability and sustainability of IoT networks, particularly those relying on battery-powered sensor nodes. Optimizing energy usage is paramount for extending the operational lifespan of these nodes, reducing maintenance costs, and minimizing environmental impact. By developing energy-efficient algorithms and protocols, the proposed scheme not only enhances the longevity of sensor nodes but also contributes to the overall resilience and affordability of IoT deployments. In the proposed scheme, we made our assumptions regarding energy consumption due to the fixed space required for watermark embedding. The phases that affect energy consumption in a sensor node are sensor node activation cost, watermark generation and embedding cost, data capturing cost, data transmission cost and cost for going to sleeping mode. The energy (E_n) of each sensor node in the network is computed according to Eq. (3). The power (P_n) utilized by each node is determined by node's hardware components, the network's data rate, and the communication protocols used by the network. In order to estimate the power consumption of the sensor node for numerical simulation we use the energy model in [49] based on Mica2 Motes. The time to complete a round of a sensor node operation specified in Fig. 13 is T_n which varies according to the data processing method and functionality of this node as shown in the figure. We assume, as in [49], that the parameters used in the energy calculation are as follows: $T_A = 1$ ms (Active time cost), $T_S = 0.5$ ms (Data sensing time), T_C (Computation and processing time), $T_{TR} = 300$ ms (Data transmission time), $T_{SL} = 299$ ms (sleeping time cost), and $P_n = 30$ mW (Average power consumption of a single sensor node). Using Eq. (5) we compute the energy of sensor nodes based on the previously specified parameters.

$$E_n = P_n \times T_n, \tag{4}$$

$$E_n = P_n \times (T_A + T_S + T_C + T_{TR} + T_{SL}). \tag{5}$$



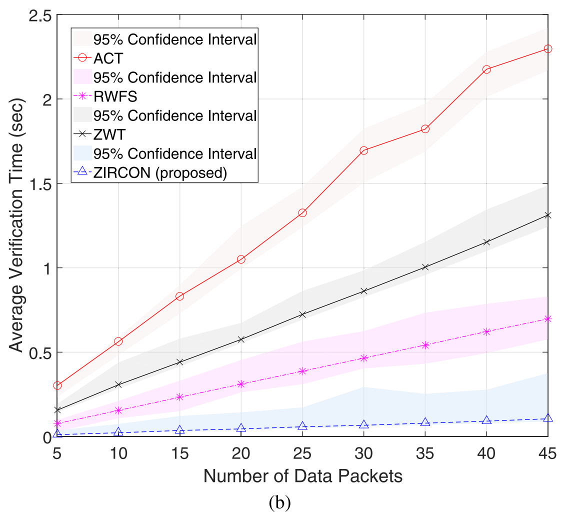


Fig. 10. Computational time. (a) Watermark generation and embedding time. (b) Watermark verification time.

The analysis of the energy consumption of ZIRCON scheme compared to RWFS [32], ACT [30] and ZWT [35] shows that our approach requires less energy for each operating node. This results in an increase in life time of our network compared to other networks. The higher energy consumption in RWFS [32] is based on the computation of bit positions for watermark embedding and the encryption of captured data (that is used as an input to an HMAC function to obtain a final watermark) using homomorphic encryption algorithm. This method is slower than conventional symmetric encryption methods because of the complex mathematics it requires. In comparison to homomorphic encryption, symmetric encryption is quicker and easier to use because uses a single key to encrypt and decrypt data. Also, in ACT [30] the use of asymmetric cryptography functions and group hashing requires more energy at each sensor node due to the additional computational overhead required for the public and private key operations. The existing scheme ZWT [35], which uses DES for watermark encryption, dissipates higher energy than the proposed scheme which uses AES for sub-watermark encryption. Figs. 11 and 12 shows the energy consumption of the proposed scheme compared to existing state-of-the-art methods RWFS [32], ACT [30] and ZWT [35], for a single source node and an intermediate node respectively. It is clearly shown that ZIRCON requires less energy consumption at each node of the network.

5.2. Cost analysis

Regarding cost analysis, we compare ZIRCON with three state-of-the-art methods in terms of transmission data size and data packet length. Transmission data size refers to the amount of data transferred over the network for each communication session or operation. It assesses the efficiency of data transmission in IoT networks by quantifying the volume of information exchanged between nodes. It accounts for both payload data and any additional metadata, such as provenance information or watermarks. Understanding transmission data size is important for optimizing network bandwidth usage and resource allocation. Smaller data sizes reduce transmission overhead, leading to faster communication, reduced latency, and improved network scalability. Moreover, minimizing data size conserves energy and extends the battery life of constrained IoT devices, enhancing overall network sustainability and operational efficiency. The second metric is packet length which refers to the size or length of individual data packets transmitted within the network. It evaluates the granularity of data transmission and the size of individual units of information exchanged between nodes. It includes factors such as payload size, header information, and any additional protocol-specific overhead. Optimizing packet length helps minimize transmission delays, enhance real-time

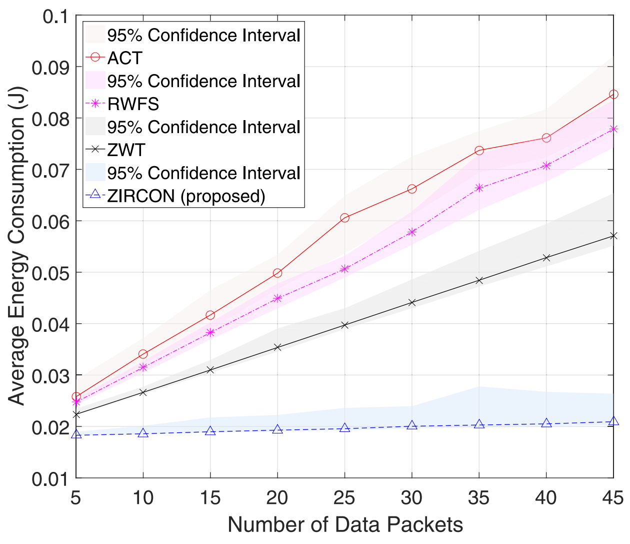


Fig. 11. Energy consumption cost per single source node.

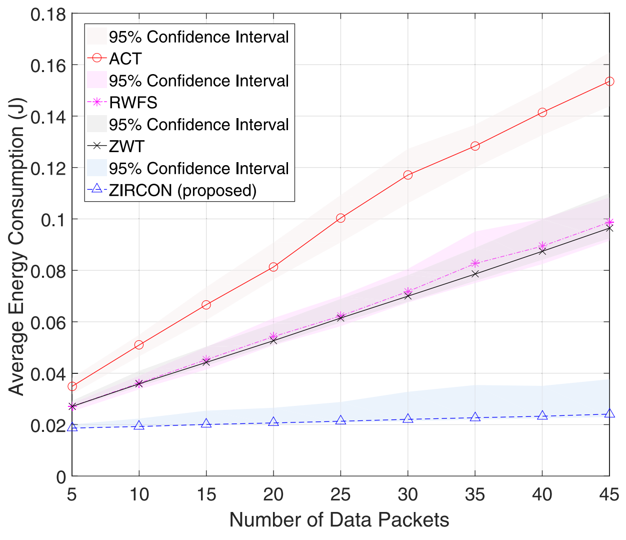


Fig. 12. Energy consumption cost per single intermediate node.

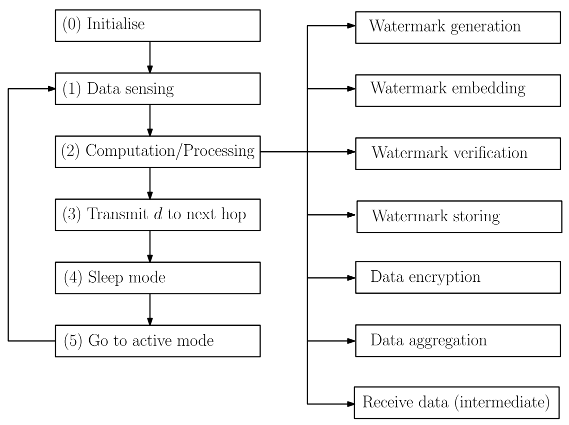


Fig. 13. Sensor node operation cycle. The initialization phase is represented as step 0 in the cycle, which is not included in the performance evaluation of computational time and energy usage of a node. The processes on the right side indicate various computation and processing tasks that a sensor node may perform.

responsiveness, and facilitate efficient use of network resources, making it essential for building robust and scalable IoT infrastructures. The state-of-the-art approaches for cost analysis are as follows:

1. The secure provenance framework $SProv$ [44] that is adapted to sensor networks by [20]. The provenance record at a node n_i is $p_i =< n_i, \text{hash}(D_i), C_i >$, where $\text{hash}(D_i)$ is a one way hash function of the updated data and $C_i = \text{sign}(\text{hash}(n_i, \text{hash}(D_i) \mid C_{i-1}))$ is an integrity checksum. This method is referred to as SSP.
2. The MAC-based provenance scheme which computes a MAC value and send it with the node ID as the provenance record. This method is referred to as MP [20].
3. A lightweight secure scheme BFP [20] that uses Bloom Filters to encode provenance information, which is sent along the data path with the data packet.

In the proposed scheme, a sensor node transmits both provenance information (IP address, timestamp, and sequence number) and a hash value as a zero-watermark. The IP address, packet timestamp and sequence number have a size of 4 bytes each. The source node encrypts the sub-watermark $sw_{f_{n,k,i}}$ and produce an encrypted sub-watermark of 16 bytes. Also, the source node computes the hash value from the extracted data payload, as shown in Algorithm 1, and selects the first 8 bytes. This implies that the generated zero-watermark including the provenance record is 24 bytes. The provenance record is stored in a tamper-proof network database at each hop. Hence, each data packet holds only one generated zero-watermark in each hop. For SSP, to perform cryptographic hash operations, they utilize SHA-1 with a bit length of 160, and for generating digital signatures of 160 bits (ECDSA), they make use of the TinyECC library [50]. The node ID, which is 2 bytes long, results in each provenance record being 42 bytes in length. To implement MP, the provenance record is formed of node ID and a MAC value computed on each source node. It uses the TinySec library [51] to compute the sensor CBC-MAC of size 4 bytes. Thus, the provenance record is of 6-byte size. In both schemes, SSP and MP, each node embeds its provenance information as a record with data packet, as the path length increases the provenance size increases linearly. This increase in provenance leads to an increase in the transmitted data packet size. In a multi-hop scenario, the provenance is $6 \times H$ bytes (i. e., the path is formed of H hops) for MP and $42 \times H$ bytes for SSP. However, in BFP, the provenance length depends on parameter selection of the Bloom Filter. For a given H and a false positive probability $P_{fp} = 0.02$, the number of required bits to encode the provenance information is $m = (-H \cdot \ln(P_{fp}))/(\ln 2)^2$. In this case the length of a BF grows with the number of nodes. Fig. 14(a) shows a comparison between ZIRCON, SSP, MP and BFP approaches in-terms of transmission data size in a single hop scenario. Similarly, the results for data packet length in a Multi-hop scenario for both schemes is shown in Fig. 14(b). In resource constrained networks, energy is mainly affected by data transmission, which increases as the data packet increase. The results show that ZIRCON performs better than SSP and MP as the number of hops increases in the sensor network. Also, our algorithm outperforms BFP in terms of provenance length and scalability as the size of the network increases, and as the number of hops exceeds 11. the proposed model only encodes one provenance record $p_{n,k,i}$ with each data packet $d_{n,k}$ during transmission.

6. Discussion

The related literature includes many proposed schemes for ensuring data integrity and secure provenance transmission in WSNs using digital watermarking. These models are elaborated in Section 2. The limitations of such solutions were addressed in the proposed scheme. In this scheme we combine both data integrity and secure provenance transmission, taking into consideration the computational capabilities of sensor nodes in IoT networks, while maintaining security standards. IoT networks are vulnerable to many type of attacks. These networks are used in decision making processes that require high level of security. Moreover, it is essential in many situations to keep track of the data captured from sensor nodes to identify any malicious traffic in the context of intrusion detection systems. For this, it requires to overcome a set of challenges in order to securely transmit provenance information.

The difficulties involve handling processing overhead of each network node, transmitting information about the origin of data in an efficient way without using extra bandwidth and quickly responding to any security breaches. Provenance information grows very fast, which requires transmitting large amount of provenance information with data packets. In fact, building the lineage of each data-packet requires storing the information of the data-packet including the complete set of nodes that were covered from source to destination. Embedding such vast amount of information with the data packet will result in a massive network overhead. This requires a solution for handling this amount of provenance information. This critical problem was not addressed in the related literature. In this context, we propose the use of a tamper-proof database to store these information that are embedded in watermarks at each node covered in the network. Hence, to obtain the required security standards, the proposed zero-watermarking approach generates two sub-watermarks that are used for integrity verification and secure provenance transmission. The sub-watermarks are based on one-way hash function (i. e., SHA-2) and symmetric encryption (i. e., AES). In our work, we provide an efficient and secure way to keep track of the whole network route that a piece of information has taken despite bandwidth overhead, storage limitations and computational overhead, while ensuring data integrity.

Managing internal data packets is a key component of improving IDS efficiency. Analyzing each data packet by the IDS at each node implies additional computational overhead. This issue was not addressed in the related literature, which only focus on data packets that are specified for sensed data. In our model, we propose a protocol for labeling internal managing data packets which allows to check for any attack at the level of these packets without the need to analyze it by IDS. In our work, we validate our zero-watermarking algorithm through a security analysis that shows our approach is robust to many attacks based on an attack model. Additionally, we provide a performance evaluation to analyze computational time, energy consumption and cost analysis in comparison with related literature. As a result of our security scheme outlined and proposed in this paper, there are several areas for future study and improvement. The fast evolution of security attacks against IoT networks, such as Distributed Denial of Service (DDoS), Botnets, Privacy invasion and Physical attack, requires the

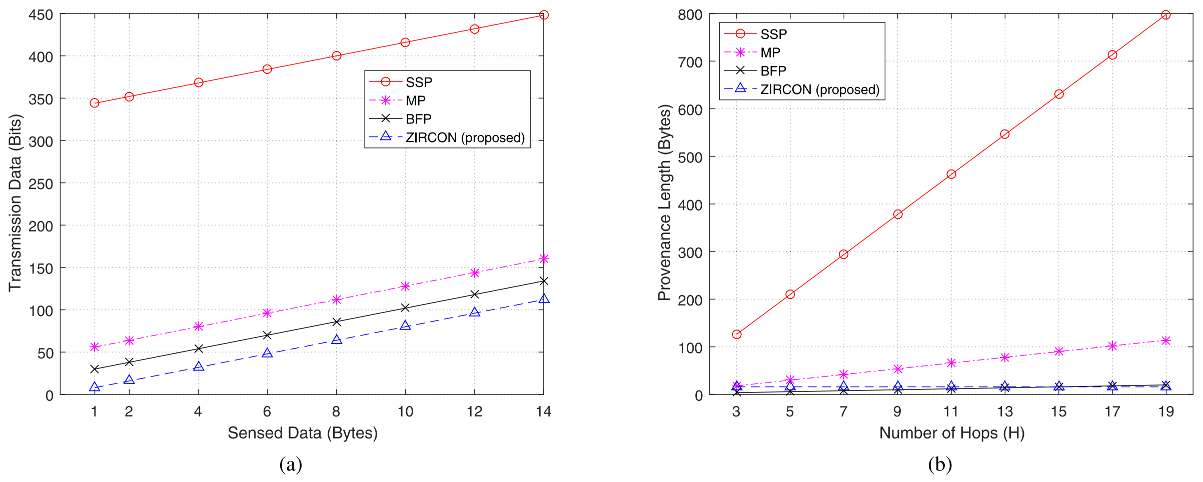


Fig. 14. Cost comparison. (a) Transmission data size in the single hop scenario. (b) Provenance length in the multi-hop scenario.

advancement in security measures to protect against these types of attacks and to ensure the security of IoT networks. This presents an important call to discover ways to tackle the vast number of security attacks that are not yet studied in the area of securing data provenance in integrity in IoT networks. Another issue is that large-scale IoT networks that introduce the problem of large-scale provenance need to be analyzed. How to handle the huge lineage of data being transmitted over long data-path. Even in the presence of a database, how to manage methods to efficiently overseeing a large quantity of sensor nodes and the data they collect, along with information about its origin and the path it covers.

7. Conclusion

This paper addresses the problem of data integrity and secure transmission of provenance information for IoT networks. We propose a zero-watermarking approach that embeds provenance information and data features with data packets and stores these watermarks in a tamper-proof network database. The security capabilities of ZIRCON make it secure against different types of sensor network attacks, as proved using a formal security analysis. We have validated our findings by conducting representative simulations and compared our results with existing schemes based on different performance parameters. The results show that the proposed scheme is lightweight, has better computational efficiency, and consumes less energy, compared to prior art. Perspectives for future work include testing this scheme against various attacks that are not included in our threat model and studying the possibility of using this method for large-scale provenance.

CRediT authorship contribution statement

Omair Faraj: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **David Megías:** Conceptualization, Formal analysis, Funding acquisition, Methodology, Project administration, Resources, Supervision, Writing – review & editing. **Joaquin Garcia-Alfaro:** Conceptualization, Formal analysis, Methodology, Project administration, Resources, Supervision, Writing – review & editing.

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.

Data availability

Data will be made available on request.

Acknowledgments

The authors acknowledge the funding obtained from the EIG CONCERT-Japan with grant PCI2020-120689-2 (Government of Spain) “DISSIMILAR” and the PID2021-125962OB-C31 “SECURING” project granted by the Spanish Ministry of Science and Innovation, as well as the ARTEMISA International Chair of Cybersecurity (C057/23) and the DANGER Strategic Project of Cybersecurity (C062/23), both funded by the Spanish National Institute of Cybersecurity through the European Union – NextGenerationEU and the Recovery, Transformation and Resilience Plan. The authors acknowledge as well support from IMT’s CyberCNI chair (Cybersecurity for Critical Networked Infrastructures, cf. https://cybercni.fr/).

References

[1] Ammar M, Russello G, Crispo B. Internet of Things: A survey on the security of IoT frameworks. J Inf Secur Appl 2018;38:8–27. http://dx.doi.org/10.1016/j.jisa.2017.11.002.

[2] Botta A, de Donato W, Persico V, Pescapé A. Integration of Cloud computing and Internet of Things: A survey. Future Gener Comput Syst 2016;56:684–700. http://dx.doi.org/10.1016/j.future.2015.09.021.

[3] Gubbi J, Buyya R, Marusic S, Palaniswami M. Internet of Things (IoT): A vision, architectural elements, and future directions. Future Gener Comput Syst 2013;29(7):1645–60. http://dx.doi.org/10.1016/j.future.2013.01.010.

[4] Lazarescu MT. Wireless sensor networks for the internet of things: Barriers and synergies. In: Components and services for IoT platforms: paving the way for ioT standards. Springer International Publishing; 2017, p. 155–86. http://dx.doi.org/10.1007/978-3-319-42304-3_9.

[5] Manikandan N, Subha S. Parallel AES algorithm for performance improvement in data analytics security for IoT. Int J Netw Virtual Organ 2018;18(2):112–29. http://dx.doi.org/10.1504/IJNVO.2018.091575.

[6] Sicari S, Rizzardi A, Grieco L, Coen-Porisini A. Security, privacy and trust in Internet of Things: The road ahead. Comput Netw 2015;76:146–64. http://dx.doi.org/10.1016/j.comnet.2014.11.008.

[7] Zarpelão BB, Miani RS, Kawakani CT, de Alvarenga SC. A survey of intrusion detection in Internet of Things. J Netw Comput Appl 2017;84:25–37. http://dx.doi.org/10.1016/j.jnca.2017.02.009.

[8] Ren K, Lou W, Zhang Y. LEDS: Providing location-aware end-to-end data security in wireless sensor networks. IEEE Trans Mob Comput 2008;7(5):585–98. http://dx.doi.org/10.1109/TMC.2007.70753.

[9] Khan MA, Salah K. IoT security: Review, blockchain solutions, and open challenges. Future Gener Comput Syst 2018;82:395–411. http://dx.doi.org/10.1016/j.future.2017.11.022.

[10] Sultana S, Shehab M, Bertino E. Secure provenance transmission for streaming data. IEEE Trans Knowl Data Eng 2013;25(8):1890–903. http://dx.doi.org/10.1109/TKDE.2012.31.

[11] Dai C, Lin D, Bertino E, Kantarcioglu M. An approach to evaluate data trustworthiness based on data provenance. In: Jonker W, Petković M, editors. Secure data management. Berlin, Heidelberg: Springer Berlin Heidelberg; 2008, p. 82–98. http://dx.doi.org/10.1007/978-3-540-85259-9_6.

[12] Van Schyndel RG, Tirkel AZ, Osborne CF. A digital watermark. In: Proceedings of 1st international conference on image processing. Vol. 2, IEEE; 1994, p. 86–90. http://dx.doi.org/10.1109/ICIP.1994.413536.

[13] Fei C, Kundur D, Kwong RH. Analysis and design of secure watermark-based authentication systems. IEEE Trans Inf Forensics Secur 2006;1(1):43–55. http://dx.doi.org/10.1109/TIFS.2005.863505.

[14] Cox I, Miller M, Bloom J, Fridrich J, Kalker T. Digital watermarking and steganography. Morgan Kaufmann Publishers Inc.; 2007, http://dx.doi.org/10.1016/B978-0-12-372585-1.X5001-3.

[15] Megías D. Data hiding: New opportunities for security and privacy? In: Proceedings of the European interdisciplinary cybersecurity conference. EICC 2020, 2021, p. 1–6. http://dx.doi.org/10.1145/3424954.3425511.

[16] Perez-Freire L, Comesana P, Troncoso-Pastoriza JR, Perez-Gonzalez F. Watermarking security: a survey. In: Transactions on data hiding and multimedia security I. Springer; 2006, p. 41–72, URL https://dl.acm.org/doi/10.5555/2172238.2172240.

[17] Zhang G, Kou L, Zhang L, Liu C, Da Q, Sun J. A new digital watermarking method for data integrity protection in the perception layer of IoT. Secur Commun Netw 2017;2017. http://dx.doi.org/10.1155/2017/3126010.

[18] Kamal M, Tariq M. Light-weight security and data provenance for multi-hop internet of things. IEEE Access 2018;6:34439–48. http://dx.doi.org/10.1109/ACCESS.2018.2850821.

[19] Sultana S, Bertino E, Shehab M. A provenance based mechanism to identify malicious packet dropping adversaries in sensor networks. In: 2011 31st international conference on distributed computing systems workshops. 2011, p. 332–8. http://dx.doi.org/10.1109/ICDCSW.2011.54.

[20] Sultana S, Ghinita G, Bertino E, Shehab M. A lightweight secure scheme for detecting provenance forgery and packet drop attacks in wireless sensor networks. IEEE Trans Dependable Secure Comput 2015;12(3):256–69. http://dx.doi.org/10.1109/TDSC.2013.44.

[21] Suhail S, Hussain R, Abdellatif M, Pandey SR, Khan A, Hong CS. Provenance-enabled packet path tracing in the RPL-based internet of things. Comput Netw 2020;173:107189. http://dx.doi.org/10.1016/j.comnet.2020.107189.

[22] Liu Z, Wu Y. An index-based provenance compression scheme for identifying malicious nodes in multihop IoT network. IEEE Internet Things J 2020;7(5):4061–71. http://dx.doi.org/10.1109/JIOT.2019.2961431.

[23] Siddiqui MS, Rahman A, Nadeem A. Secure data provenance in IoT network using bloom filters. Procedia Comput Sci 2019;163:190–7. http://dx.doi.org/10.1016/j.procs.2019.12.100, 16th Learning and Technology Conference 2019Artificial Intelligence and Machine Learning: Embedding the Intelligence.

[24] Aman MN, Basheer MH, Sikdar B. A lightweight protocol for secure data provenance in the internet of things using wireless fingerprints. IEEE Syst J 2021;15(2):2948–58. http://dx.doi.org/10.1109/JSYST.2020.3000269.

[25] Guo H, Li Y, Jajodia S. Chaining watermarks for detecting malicious modifications to streaming data. Inform Sci 2007;177(1):281–98. http://dx.doi.org/10.1016/j.ins.2006.03.014.

[26] Kamel I, Juma H. Simplified watermarking scheme for sensor networks. Int J Internet Protocol Technol 2010;5(1/2):101–11. http://dx.doi.org/10.1504/IJIPT.2010.032619.

[27] Kamel I, Juma H. A lightweight data integrity scheme for sensor networks. Sensors (Basel, Switzerland) 2011;11:4118–36. http://dx.doi.org/10.3390/s110404118.

[28] Wang B, Kong W, Xiong N. A dual-chaining watermark scheme for data integrity protection in internet of things. Comput Mater Contin 2019;58:679–95. http://dx.doi.org/10.32604/cmc.2019.06106.

[29] Shi X, Xiao D. A reversible watermarking authentication scheme for wireless sensor networks. Inform Sci 2013;240:173—-183. http://dx.doi.org/10.1016/j.ins.2013.03.031.

[30] Sun X, Su J, Wang B, Liu Q. Digital watermarking method for data integrity protection in wireless sensor networks. Int J Secur Appl 2013;7(4):407–16, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84883411518&partnerID=40&md5=1f268560dc663749dbfb41b10033b764.

[31] Zhou L, Zhang Z. A secure data transmission scheme for wireless sensor networks based on digital watermarking. In: 2012 9th international conference on fuzzy systems and knowledge discovery. 2012, p. 2097–101. http://dx.doi.org/10.1109/FSKD.2012.6234194.

[32] Alromih A, Al-Rodhaan M, Tian Y. A randomized watermarking technique for detecting malicious data injection attacks in heterogeneous wireless sensor networks for internet of things applications. Sensors (Basel, Switzerland) 2018;18(12). http://dx.doi.org/10.3390/s18124346.

[33] Lalem F, Muath A, Bounceur A, Euler R, Laouamer L, Nana LT, et al. Data authenticity and integrity in wireless sensor networks based on a watermarking approach. In: The 29th international florida artificial intelligence research society. FLAIRS-29, Key Largo, Florida, United States; 2016, p. 276–81, URL https://hal.univ-brest.fr/hal-01294146.

[34] Soderi S. Acoustic-based security: A key enabling technology for wireless sensor networks. Int J Wirel Inf Netw 2020;27(1):45–59. http://dx.doi.org/10.1007/s10776-019-00473-4.

[35] Hameed K, Khan A, Ahmed M, Reddy AG, Rathore MM. Towards a formally verified zero watermarking scheme for data integrity in the Internet of Things based-wireless sensor networks. Future Gener Comput Syst 2018;82:274–89. http://dx.doi.org/10.1016/j.future.2017.12.009.

[36] Boubiche D, Boubiche S, Toral-Cruz H, Pathan A-SK, Bilami A, Athmani S. SDAW: Secure data aggregation watermarking-based scheme in homogeneous WSNs. Telecommun Syst 2015;59:277–88. http://dx.doi.org/10.1007/s11235-015-0047-0.

[37] Park U, Heidemann J. Provenance in sensornet republishing. In: Provenance and annotation of data and processes: second international provenance and annotation workshop, IPAW 2008, Salt Lake City, UT, USA, June 17-18, 2008. revised selected papers. Berlin, Heidelberg: Springer-Verlag; 2008, p. 280–92. http://dx.doi.org/10.1007/978-3-540-89965-5_28.

[38] Lim H-S, Moon Y-S, Bertino E. Provenance-based trustworthiness assessment in sensor networks. In: Proceedings of the seventh international workshop on data management for sensor networks. DMSN '10, New York, NY, USA: Association for Computing Machinery; 2010, p. 2–7. http://dx.doi.org/10.1145/1858158.1858162.

[39] Bertino E. Security and privacy in the IoT. Springer, Cham; 2018, p. 3–10. http://dx.doi.org/10.1007/978-3-319-75160-3_1.

[40] Monteiro N, Mihovska A, Rodrigues A, Prasad N, Prasad R. Interference analysis in a LTE-a HetNet scenario: Coordination vs. uncoordination. In: Wireless VITAE 2013. IEEE; 2013, p. 1–5.

[41] Cui J, Shao L, Zhong H, Xu Y, Liu L. Data aggregation with end-to-end confidentiality and integrity for large-scale wireless sensor networks. Peer-to-Peer Netw Appl 2018;11(5):1022–37. http://dx.doi.org/10.1007/s12083-017-0581-5.

[42] Ganesan P, Venugopalan R, Peddabachagari P, Dean A, Mueller F, Sichitiu M. Analyzing and modeling encryption overhead for sensor network nodes. In: Proceedings of the 2nd ACM international conference on wireless sensor networks and applications. WSNA '03, New York, NY, USA: Association for Computing Machinery; 2003, p. 151–9. http://dx.doi.org/10.1145/941350.941372.

[43] Roosta TG. Attacks and defenses of ubiquitous sensor networks (Ph.D. thesis), USA: University of California at Berkeley; 2008, URL https://dl.acm.org/doi/book/10.5555/1559508, AAI3331772.

[44] Hasan R, Sion R, Winslett M. The case of the fake picasso: Preventing history forgery with secure provenance. In: 7th USeNIX conference on file and storage technologies (FAST 09). San Francisco, CA: USENIX Association; 2009, URL https://www.usenix.org/conference/fast09/technical-sessions/presentation/hasan.

[45] Castelluccia C, Chan AC-F, Mykletun E, Tsudik G. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. ACM Trans Sen Netw 2009;5(3). http://dx.doi.org/10.1145/1525856.1525858.

[46] Halgamuge MN, Zukerman M, Ramamohanarao K, Vu H. An estimation of sensor energy consumption. Progr Electromagn Res B 2009;12(12):259–95. http://dx.doi.org/10.2528/PIERB08122303.

[47] Fu C, Jiang Z, Wei W, Wei A. An energy balanced algorithm of LEACH protocol in WSN. IJCS 2013;10(1):354–9, URL https://www.proquest.com/scholarly-journals/energy-balanced-algorithm-leach-protocol-wsn/docview/1441692218/se-2.

[48] N. Halgamuge M, Zukerman M, Ramamohanarao K, L. Vu H. An estimation of sensor energy consumption. Progr Electromagn Res B 2009;12:259–95. http://dx.doi.org/10.2528/PIERB08122303.

[49] Miller M, Vaidya N. A MAC protocol to reduce sensor network energy consumption using a wakeup radio. IEEE Trans Mob Comput 2005;4(3):228–42. http://dx.doi.org/10.1109/TMC.2005.31.

[50] Liu A, Ning P. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In: 2008 international conference on information processing in sensor networks (ipsn 2008). 2008, p. 245–56. http://dx.doi.org/10.1109/IPSN.2008.47.

[51] Karlof C, Sastry N, Wagner D. TinySec: A link layer security architecture for wireless sensor networks. In: Proceedings of the 2nd international conference on embedded networked sensor systems. SenSys '04, New York, NY, USA: Association for Computing Machinery; 2004, p. 162–75. http://dx.doi.org/10.1145/1031495.1031515.