Pervasive and Mobile Computing 99 (2024) 101889

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

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

A novel IoT trust model leveraging fully distributed behavioral fingerprinting and secure delegation

Marco Arazzi [a], Serena Nicolazzo [b], Antonino Nocera [a,*]

[a] Department of Electrical, Computer and Biomedical Engineering, University of Pavia, via A. Ferrata, 5 - 27100 - Pavia, Italy
[b] Department of Computer Science, University of Milan, Via G. Celoria, 18 - 20133 - Milan, Italy

ARTICLE INFO

Keywords:
Internet of Things
Consensus
Blockchain
Autonomy
Reliability
Trust
Reputation

ABSTRACT

The pervasiveness and high number of Internet of Things (IoT) applications in people's daily lives make this context a very critical attack surface for cyber threats. The high heterogeneity of involved entities, both in terms of hardware and software characteristics, does not allow the definition of uniform, global, and efficient security solutions. Therefore, researchers have started to investigate novel mechanisms, in which a super node (a gateway, a hub, or a router) analyzes the interactions of the target node with other peers in the network, to detect possible anomalies. The most recent of these strategies base such an analysis on the modeling of the fingerprint of a node behavior in an IoT; nevertheless, existing solutions do not cope with the fully distributed nature of the referring scenario.

In this paper, we try to provide a contribution in this setting, by designing a novel and fully distributed trust model exploiting point-to-point devices' behavioral fingerprints, a distributed consensus mechanism, and Blockchain technology. In our solution we tackle the non-trivial issue of equipping smart things with a secure mechanism to evaluate, also through their neighbors, the trustworthiness of an object in the network before interacting with it. Beyond the detailed description of our framework, we also illustrate the security model associated with it and the tests carried out to evaluate its correctness and performance.

1. Introduction

Nowadays, the Internet of Things (IoT) has grown rapidly, attracting not only researchers but also people from industrial and commercial environments. Indeed, this paradigm, characterized by heterogeneous and connected devices sharing data and providing services, creates huge opportunities in numerous domains. Radio Frequency identification (RFID), wireless sensors and other smart technologies are integrated into a variety of applications to create networks with enhanced capabilities in terms of sensing information about the environment and collecting measurements or operational data from their devices.

One of the peculiarities of this new scenario is that there is no need for constant human intervention for the *things* to handle data, process them, exchange messages in the network, or execute instructions [1,2]. A typical case, in which the user takes almost no active role and fully relies on devices and services to act on her/his behalf, is the smart home environment. Think, for instance, of sensors able to recognize the presence of humans in home rooms to switch on/off lights. The information collected and used to perform this task is one of the most sensitive and personal (i.e., people's movements inside their own homes) [3]. Hence, solving the trade-off between the functionalities provided (as well as the degree of autonomy of the objects) and the sensitivity of data exchanged to obtain them is a demanding challenge to be faced. Moreover, the need for privacy increases when an untrusted (or

* Corresponding author.
E-mail addresses: marco.arazzi01@universitadipavia.it (M. Arazzi), serena.nicolazzo@unimi.it (S. Nicolazzo), antonino.nocera@unipv.it (A. Nocera).

https://doi.org/10.1016/j.pmcj.2024.101889
Received 10 August 2022; Received in revised form 19 October 2023; Accepted 1 February 2024
Available online 8 February 2024

also honest-but-curious) party is involved, and, in the aforementioned scenario, data is not completely processed inside the user's object directly, but rather it is handled by some external cloud-based service.

Other domains in which lots of sensitive information is exchanged involve healthcare [4,5]. In this case, wearable devices collect data about users' health conditions continuously to monitor them for medical security reasons. Also, in this case, data loss or compromise can result in serious damages [6]. Moreover, if an IoT device is implanted in a human body for the good functioning of an organ, an attacker gaining access to it can endanger the life of the victim. Unfortunately, the hacking of a device is not a remote possibility, indeed, a research study carried out by Packard, reported that more than 70% of the existing IoT systems have severe vulnerabilities due to a number of reasons, such as insecure Web interfaces, insufficient authorization mechanisms, lack of encryption for transport, and inadequate software protection [7,8].

To make reference to recent events, think for instance of famous botnet attacks launched against a large number of IoT devices, such as: the Mirai attack in 2016, that infected around 2.5 million devices connected to the Internet through a Distributed Denial of Service (DDoS) attack, and the Hajime attack, which brings more sophistication to some of the techniques used by Mirai [9,10].

Another domain gaining benefits from the growth of IoT is the Cyber–Physical System (CPS, hereafter) context. In such systems, physical and software components are deeply interconnected to continuously monitor the environment, and thanks to the support of intelligent systems, CPSs are also able to make decisions based on the physical changes in the surroundings. Since CPSs control assets of critical importance (e.g., industrial control systems, power grids, transportation), neglecting security standards can lead to serious consequences.

Besides autonomy, a factor that makes the management of IoT devices even more critical is their intrinsic heterogeneity. Their different computational capabilities, memory, and provided functionalities, cause that, generally, poorly secured objects can connect to the Internet and can interact with more powerful devices. Hence, these objects can represent multiple points of failure that expose the entire IoT network to possible attacks, increasing the need to define and adopt non-standard security methods [11].

In this scenario, the classical countermeasures to face privacy and security threats have to be rethought taking into account the many restrictions and limitations of IoT devices (in terms of components and devices, computational, and power resources) and even their heterogeneous and distributed nature. Since IoT technologies and applications are so intimately associated with people, a step forward in this direction could make consumers less reluctant to adopt this new paradigm.

Recently, researchers have started to exploit the possibility for nodes to collaborate to make IoT networks more robust to attacks [2,12]. One area of investigation in this context is trust and reputation management, which is crucial to allow efficient collaboration among the actors of the network that might not have sufficient prior knowledge about each other [13]. Trust in IoT is a comprehensive concept that takes people, devices, and their connections into account. It can be defined as the expectation that a thing is reliable, in other words, that it acts without harming the user or other objects in the network, is resilient to attacks, and belongs to a user who is always who she/he claims to be [14]. To make an example, assessing that a smartwatch is trusted could mean that it always gives the right (i.e., correct, complete, and fresh) information to its user when queried (for instance, time, temperature, heartbeat, and so forth). Moreover, by doing so, it should not reveal more information about user habits than necessary, and it should not act as a malicious node performing tasks it is not authorized for.

As for thing-to-thing trust evaluation, classical approaches exploit remote attestation. Through this security service, an object can evaluate the current state of a potentially hacked or compromised remote device before contacting it. Remote attestation algorithms span from heavy-weight secure hardware-based techniques (such as cryptography), to light-weight software-based ones (e.g., control-flow integrity) [15].

More recent approaches are based on the computation of device *fingerprint* [16,17]. Fingerprint represents a set of features useful to identify an object not relying on its classical network identities (such as IP or MAC addresses), but exploiting the information from the packets that the device exchanges over the network. Always in this context, a step forward is represented by the notion of *behavioral fingerprint*. All the approaches based on this concept build a device profile describing what it usually does (how it interacts with the environment) and some patterns of communication (both with other objects and humans). This profile is used to analyze the current behavior of an object and to assess whether it is congruent with the expected one [18,19].

However, classical trust models are usually too computationally heavy and do not exploit the peculiarity of IoT nodes that can collaborate to obtain consensus. Moreover, most of the approaches based on behavioral fingerprint are centralized and, as a consequence, do not take into consideration features obtainable by analyzing message payloads [16].

We start from the above considerations to design a solution based on a fully distributed behavioral fingerprint computation used as an input to a general IoT trust model. Essentially, to assess the trustworthiness of an object in the network our approach proceeds through two steps. The former is the construction of behavioral models representing the expected conduct of every node in the network, and the latter is a suitable monitoring activity to detect possible variations in it.

The novelty and importance of our solution in the context of pervasive computing lays also on the fact that to reach our goal, we exploit all the peculiarities of IoT nodes, such as their being autonomous, heterogeneous, and collaborative. Indeed, in our approach, we enforce that:

- To guarantee objects autonomy we design a solution in which nodes can learn models to represent the expected behavior of a target object unobtrusively without human intervention.
- To exploit the intrinsic heterogeneous nature of IoT objects, the training and inference phases can be also obtained through a privacy-preserving collaborative delegation approach in which simpler objects are supported by more powerful nodes to implement the solution.

- Finally, since collaboration is a fundamental characteristic of a network of things, we based our anomaly detection algorithm on it. Indeed, nodes cooperate to constantly monitor other objects' conduct and signal possible anomalies in their normal behavior through a distributed algorithm based on a consensus mechanism.

Another point of strength of our approach is that it is completely distributed. Blockchain technology is used to deploy part of our solution in a fully decentralized manner. In particular, we exploit both the power of smart contracts, which are already being used to manage, control, and secure IoT devices [20], and a lightweight adaptation of Blockchain designed to support resource-constrained IoT devices [2]. Relying on these solutions our approach allows to: *(i)* keep trace of the evaluation of the behavior of objects at a global level and *(ii)* identify the best peers to contact to enable the aforementioned collaborative approach.

In the following of this paper, through a deep experimental campaign, carried out leveraging real-life smart object data, we prove that our approach is feasible and equips the nodes of an IoT network with the possibility to detect if another peer is compromised before contacting it. Interestingly, our strategy is based on a lightweight behavioral fingerprint model suitable for IoT devices and our secure delegation strategy produces advantages also in terms of running time.

Our paper starts from the research direction described in [18], where the framework H2O (Human to Object) is presented. Node belonging to H2O can continuously authenticate an entity in the network, providing a reliability assessment mechanism based on behavioral fingerprinting. Our proposal extends that work by presenting several critical enhancements and designing novel security mechanisms to improve object interaction in IoT. In particular, the contributions of our paper can be summarized as follows.

1. We extend the ideas proposed in the recent scientific literature [19,21] by developing a light deep learning model for the computation of the distributed behavioral fingerprinting also through a Tiny Machine Learning approach. This improves the usability of our solution across several device types, even the less computationally capable ones.
2. We propose a distributed consensus mechanism and design a novel secure delegation strategy to compute the object's reliability. This allows the improvement of the security and autonomy of smart objects.
3. We add a Blockchain-based solution to trace, in a fully distributed fashion, the evolution of the behavior of objects when interacting with each other.
4. Through a detailed security analysis, we show that our proposal is robust and addresses its objectives in the presence of attacks.

The outline of this paper is as follows. In Section 2, we examine the literature related to our approach. In Section 3, we give a general overview of our reference IoT model and illustrate the proposed framework in detail. In Section 4, we describe our security model. In Section 5, we present the set of experiments carried out to test our approach and analyze its performance. Finally, in Section 7, we draw our conclusions and have a look at possible future developments of our research efforts.

2. Related works

Since the IoT environment is widely distributed and dependable on user-sensitive data, the concept of trust management is becoming a crucial prerequisite for the design of new applications in this field [14]. In scientific literature, different trust definitions have been settled but, speaking about IoT devices, the research community agrees to define trust as the probability that the intended behavior of a thing and its actual behavior are equivalent, given fixed context, environment, and time [13].

In the last years, different approaches for trust management of IoT object-to-object communications have been provided. Typically, the mechanism through which a node can check the current state of a potentially compromised remote device, before interacting with it, is referred to as *remote attestation*. This security service is implemented by [22], where a Multiple-Tier Remote Attestation protocol, called MTRA, is presented. In particular, this framework provides two methods to monitor IoT devices on the basis of their characteristics. Specifically, less smart devices are verified through a lighter software-based attestation algorithm, whereas more powerful ones are monitored by means of trusted hardware called Trusted Platform Module (TPM). Another scheme in this context, providing enhanced functionalities is proposed in [23], where the authors describe a many-to-one attestation approach for device swarms. Through some redundancy, this solution reduces the possibility of a single point of failure typical of architectures in which a verifier node has to assess the reliability of more IoT devices.

Possible strategies to empower objects with a means to trust peers in their network are provided by cryptographic techniques [24,25]. Nevertheless, in an IoT scenario, key management represents an issue due to resource-constrained devices and a lack of a unique standard. Moreover, besides the fact that these approaches are computationally demanding, they are also not robust against internal malicious nodes having valid cryptographic keys. Finally, another weakness is that they usually rely on an external level for the computation of the node trust score.

Still in this context, some approaches aim at facing threats to reputation models, such as bad-mouthing and slandering attacks [26,27]. For instance, in [26] the authors show a trust mechanism with a consensus-based false information filtering algorithm (TM-CFIFA) to defend a wireless sensor network against bad-mouthing attacks and false-praise attacks. Whereas the authors of [27] employ trust evaluation components based on success ratio and node misbehavior to unmask possible attacks, such as on-off attacks and bad-mouthing attacks. Observe that although our approach shares some similarities with those providing countermeasures to attacks against the trust and reputation model, actually the objective of our proposal is quite different. Indeed, we aim to propose a fully distributing strategy for collaborative anomaly detection in IoT. This last task is obtained through behavioral fingerprinting models, whereas the distributed nature is supported by a secure consensus-based mechanism leveraging an underlying reputation model. In Sections 4.2 and 5.5 we show that our solution is resistant to attacks menacing the security of the reputation model (such as bad-mouthing or collusion attacks) and that our strategy can also be used to isolate nodes carrying out them.

A wide group of works focuses on the issue of IoT device identification and authentication to assess reliability, such as the strategies presented in [28,29]. They start from the consideration that network identifiers like IP addresses, MAC addresses, ports numbers, etc. have been used for identifying devices, but they can be spoofed easily.

Hence, some of them explore the concept of *device fingerprinting* as a way to identify an object not through its classical network IDs, but exploiting the information contained in the communication packets exchanged over the network. In particular, the work presented in [16] analyzes a sequence of packets from high-level network traffic to extract a set of unique flow-based features. From these features, a fingerprint for each device is created through machine learning techniques. In the same context, the authors of [30] exploit the potentialities of deep learning approaches to compute a set of features useful to provide a device fingerprint. Whereas, in [31] a framework called IoT Sentinel is described. This schema is able to automatically provide an anomaly detection task, identifying vulnerable devices being connected to an IoT network and enforcing mitigation measures for them. In this way, it can minimize damage resulting from their forgery. Also, the proposal presented in [17] presents an IoT device identification method that models the behavior of the network packets exchanged during communication by the objects. Some of these approaches are based on timing analysis and are designed to fingerprint specific devices that exhibit a certain behavior, e.g., probe scans for an access point.

A step forward in this context is carried out by some approaches dealing with *behavioral fingerprinting*. This type of technique focuses on more application-level features to model objects' traits, instead of relying only on the physical and link layers, as done by models using device fingerprints. Among these characteristics there are: protocols, request–response sequences, and any periodicity in the specific typology of packets along with their sizes [19]. In particular, in [32], the authors describe a distributed solution for behavioral fingerprinting in IoT exploiting a decentralized approach. They identify some network nodes, called gateways, that can monitor objects using trained classification models, thus assuring a more scalable solution. Some controller nodes, instead, are in charge of performing the training of the models. The feature vector identified contains 111 dimensions. Instead, the approach proposed in [29] is related to object reliability in a Multiple Internet of Things (MIoT), defining, only theoretically, the concept of object profile. Like our approach also this scheme is based on a consensus mechanism, but the main difference is that the reliability score is simply proportional to: *(i)* the fraction of successful transactions performed by the instances, and *(ii)* the reliability of the corresponding objects.

As stated in the Introduction, our work starts from the considerations analyzed in [18] where a framework called H2O (Human to Object) is presented. Also, nodes belonging to H2O are equipped with a mechanism to estimate the reliability of their contacts, but there are substantial differences with respect to our approach, that are worth to be detailed in the following.

The first improvement deals with the behavioral fingerprinting technique, allowing an object to assess if another one (which it usually interacts with) has been hacked or corrupted. In H2O, it leverages state-of-the-art approaches, whereas, in the present paper, we implement a custom model. This new model, through some technical improvements and the use of a Tiny Machine Learning approach, makes our solution suitable for devices with limited capabilities with only a maximum variation of 1% of the accuracy with respect to the state-of-the-art solutions (see Sections 3.3 and 5 for all the details).

Moreover, in H2O, a node can attest to the reliability of another peer if multiple confirmations of the peer trust score from neighbors exist (i.e., neighbors hold a fingerprint model of the peer) and if the mean of these scores is higher than a given threshold. In the present algorithm, instead, we improve the computation of the reliability score using Blockchain technology. Indeed, as an additional functionality, our approach also estimates the quality of the contributions of each node involved in the reliability estimation, to make our algorithm more robust to malicious false negative scores. This value affects the reliability score provided by that node and, to be globally accessible, it is stored in a Blockchain.

Another functionality provided by this paper is the design of a strategy for secure delegation to allow less capable devices to participate and benefit from the approach. This algorithm is community-oriented and privacy-preserving, instead in the H2O framework this possibility is only mentioned without developing a detailed implementation.

As for the work presented in [19], it illustrates an enhanced behavioral fingerprinting models, which considers also features derived from the analysis of packet payloads (for instance, different types of devices and their traffic characteristics). In our scenario, we are considering a more general IoT context in which also legacy devices are available. Therefore, starting from the two solutions above, we tried to lighten the architecture as much as possible, so that it could also be used by devices characterized by medium-to-low computational power and limited functionalities. Moreover, we reduce to the minimum possible complexity of the machine learning model in such a way as to directly involve the maximum possible number of nodes (see Section 3.3 for more details).

Another new and promising technology for establishing trust in IoT networks in a distributed way is Blockchain. Indeed, different proposals have been recently developed in order to provide forms of trust or authentication in an IoT network through this new technology. In particular, the work presented in [33] deals with an Obligation Chain containing obligations generated by a number of nodes, called Service Consumers. These transactions are first locally accepted by Service Providers and, then, shared with the rest of the network. This kind of framework is based on the concept of *Islands of Trust*, defined as the portion of the IoT network where trust is managed by both a full local PKI (Public Key Infrastructure) and a Certification Authority. Also, the approach in [34] relies on secure virtual zones (called bubbles) where things can identify and trust each other. These bubbles are obtained through Blockchain technology. Although Blockchain provides decentralized security and privacy, it has some drawbacks in terms of delay, energy, and computational overhead generated, that are not always suitable for most limited IoT devices. Both the works presented [2,35] try to overcome these limitations by proposing a light architecture for improving the end-to-end trust. The proposal presented in [35] is based on some gateways calculating the trust for sensor observations based on: *(i)* the data they receive from neighboring sensor

Table 1
Comparison of our approach with related ones.

Approach	Approach type	Trust	Reputation	Lightweight scheme	Secure delegation
Our approach	Fingerprint, Consensus, Delegation	x	x	x	x
[18,29]	Fingerprint, Consensus	x	x	–	–
[22,23]	Remote Attestation	x	–	–	–
[24,25,38]	Cryptographic	x	–	–	–
[16,17,28,30,31]	Fingerprint	x	–	–	–
[33–35]	Blockchain	x	x	x	–
[36]	Social network	x	x	–	–

nodes, *(ii)* the reputation of the sensor node, and *(iii)* the observation confidence. If the neighboring sensor nodes are associated with different gateway nodes, then, the gateway nodes may share the evidence with their neighboring gateway nodes to calculate the observation trust values. This architecture is not fully distributed and secure delegation is not performed, indeed, more powerful nodes are used as a gateway. Whereas the work presented in [2] proposes a two-tier Blockchain framework to increase the security and autonomy of smart objects in the IoT by implementing a trust-based protection mechanism. This work deals with the concept of communities of objects and relies on a first-tier Blockchain that is used only to record probing transactions performed to evaluate the trust of an object in another one of the same community or of a different community. Periodically, these transactions are aggregated and the obtained values are stored in the second-tier Blockchain to be globally accessed by all the communities. In our approach, Blockchain is solely used to keep track of the evaluation of the behavior of objects for the anomaly detection task and to identify the best peers to contact to enable the aforementioned collaborative approach.

A different perspective to build a trust and reputation scheme is taken by [36], in which the authors investigate the trustworthiness management in a Social Internet of Things (SIot, hereafter). An SIoT, first introduced by [37], models device interaction as social ties, allowing an object to crawl the network to find other (possibly heterogeneous) objects in an autonomous way and establish friendship relationships. In [36] the authors combine a subjective model and an objective one. In the former, each node computes the trustworthiness of its neighbors on the basis of its own experience and on the opinion of the friends in common with it. In the latter, the information about each node is distributed and stored in a distributed hash table structure, so that this information is accessible by all the nodes in the network.

In Table 1, we summarize the comparison with all the works introduced above based on the different functionalities provided by our approach, namely:

- Trust. A functionality that allows nodes in the network to assign a trust score to another node according to its behavior.
- Reputation. A functionality that allows a node in the network to compute a reliability score according to its neighbors' opinion about another node, even if they have not been in contact before.
- Light Fingerprinting. A functionality that allows the computation of behavioral fingerprinting for a node suitable for an IoT scenario, in which nodes have limited capabilities.
- Secure Delegation. A functionality according to which some computation can be entrusted to more capable devices in a privacy-preserving way.

In this table, with the letter 'x' we denote that the corresponding property is provided by the cited paper.

3. Description of our approach

3.1. General overview

In this section, we present a general overview of our approach. As stated in the Introduction, our proposal focuses on the definition of a fully distributed trust model for IoT using behavioral fingerprinting. Behavioral fingerprinting is a technique largely investigated in the scientific literature (see Section 2) to model the expected and typical conduct of an online device (typically, an IoT object) when interacting with other entities in the observed ecosystem. In traditional behavioral fingerprinting schemes, the modeling is usually performed by a centralized super-entity that can monitor and supervise objects' interactions (e.g., a network hub, an access point, or a base station) [21]. However, with explicit reference to the most recent trend of IoT, in which objects are more and more autonomous and, hence, equipped with higher computational capacities, our approach defines a solution for fully distributed behavioral fingerprinting that can, hence, be used as an input to a general IoT trust model.

According to the recent scientific literature, behavioral fingerprinting can be built by training a deep learning model on the information derived by the observed object (from which the fingerprint must be derived) and its communications. From a computational point of view, we can distinguish between two phases: *(i)* the training phase, and *(ii)* the inference phase. The former represents the most computationally expensive one and, depending on the amount of available data and the complexity of the involved model, it may require the exploitation of medium to high computationally capable devices. The inference phase, instead, is still a computationally demanding task but has far less impact when compared to the training phase. In general, it requires low to medium computationally capable devices also considering that optimizations can be applied during the training phase to obtain lighter models for the inference [39]. It is worth noticing that, in a modern IoT scenario, we can identify three categories of objects:

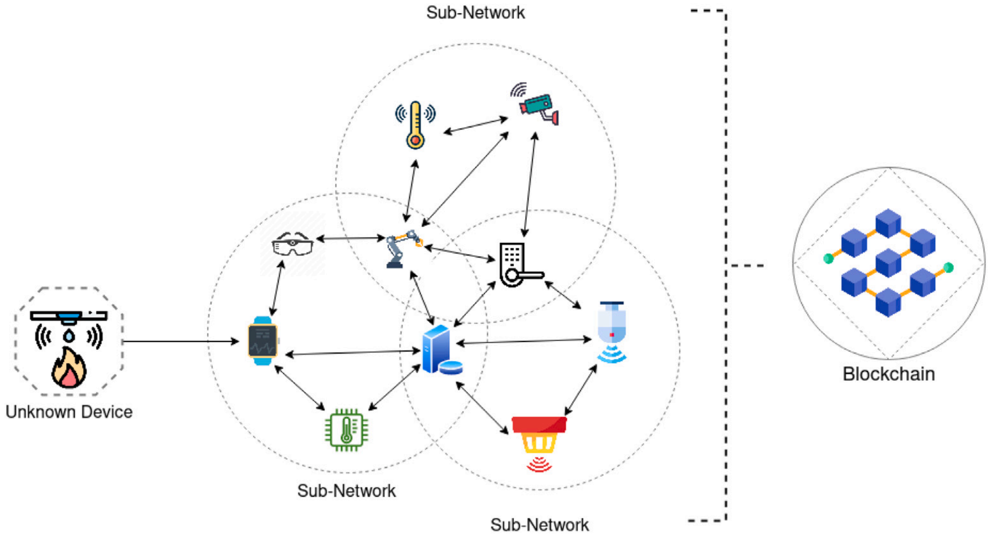

Fig. 1. The general architecture of our solution.

- Basic Device (BD, for short): low-power device with limited computational power.
- Capable Device (CD, for short): devices with sufficient computational power, but for which the training phase of the machine learning model would require too much time, or would negatively impact on battery consumption and therefore a conservative use of resources is preferable.
- Powerful Device (PD, for short): devices with high computational power and stability sufficient to both train and run machine learning models.

In general, a modern IoT is composed of a constellation of heterogeneous devices belonging to the three categories above. Therefore, for starters, to develop a fully distributed behavioral fingerprinting approach it is necessary to suitably orchestrate a collaboration mechanism to involve all the nodes in the solution. Moreover, as stated above, the goal of our approach is to design a fully distributed trust model of IoT leveraging behavioral fingerprinting as a fundamental component of an anomaly detection strategy. Therefore, information about the trustworthiness of an object in the network requires both the construction of behavioral models representing its expected conduct and suitable monitoring activities to detect possible variations in it. By leveraging a collaborative and community-based point of view, as already done by several works in the scientific literature [2,18,40,41], in our approach we enforce that more objects can learn models to represent the expected behavior of a target one and that they can collaborate to monitoring its future conduct by signaling an anomaly in case of an unexpected and impacting variation. Also, the training and inference phases can be obtained through a privacy-preserving collaborative delegation approach in which PD and CD devices cooperate and provide support to BD ones to implement the solution.

To favor interactions at a global IoT level, our approach leverages a Blockchain-based solution to support the development of the distributed trust model. In particular, a Blockchain is used to both keep track of the evaluation of the behavior of objects for the anomaly detection task and to identify the best peers to contact to enable the aforementioned collaborative approach.

In practice, our solution is composed of three main components, namely: *(i)* a distributed behavioral fingerprinting strategy; *(ii)* a community-oriented secure delegation; *(ii)* a distributed consensus mechanism to estimate object reliability.

A general architecture of our solution is reported in Fig. 1.

3.2. The proposed model

In this section, we introduce the model adopted to represent the main components and concepts exploited in our solution.

In particular, as explained above, we are considering a scenario characterized by the presence of different types of devices.

In our model, we consider the following concepts:

- the IoT node;
- the IoT interaction;
- the fingerprinting model;
- the Blockchain.

In the following, we formally define the above concepts. For the sake of clarity, in Table 2, we report all the symbols adopted in our model.

Table 2
Summary of the symbols used in our model.

Symbol	Description
D	The set of devices of the network
BD	The set of basic devices, a subset of D
CD	The set of limited devices that have sufficient computation power, a subset of D
PD	The set of powerful devices that can train machine learning models, a subset of D
d_x	A device that is part of D
p_i	A packet sent in the network
$s^t_{\langle d_x,d_y \rangle}$	The *interaction_sequence* of packets between two devices, d_x and d_y
$C_{\langle d_x,d_y \rangle}$	The set of *interaction_sequence* between d_x and d_y in D
f_{d_x,d_y}	Fingerprint model of the device d_y built by the device d_x in relation to the *communication_set* $C_{\langle d_x,d_y \rangle}$

The IoT node. IoT nodes are the main actors of our system and are associated with a profile including all the information to enable interaction with other nodes (also referred to as devices or objects). The considered profile consists of an IoT identifier and a Blockchain account. Moreover, a node includes all the information necessary to enable communication with other nodes (such as the MAC address, the IP address, and so forth). A device d_x can belong to one of the categories described in Section 3. Therefore, we can identify the following sets:

- BD: the set of basic, low-power devices.
- CD: the set of limited devices that have sufficient computation power for the inference, i.e., to use a trained machine learning model, but with limitations (e.g., battery constraints or power stability) preventing them from performing a full training.
- PD: the set of powerful devices that can train machine learning models.

More formally, we can define the set of devices of the IoT network D as:

$$D = \{d_x | d_x \in BD \cup CD \cup PD\}$$

The IoT interaction. An IoT interaction represents any data exchange between a pair of devices. In particular, an IoT interaction is established whenever a generic device, d_x, sends a packet, p_k, to another device, d_y.

Subsequently, we define an *interaction_sequence* at the time t, as a consecutive packet exchange from d_x to d_y established with a first packet p_1 sent at the time t, such that the inter-arrival time between the packages is lower than a threshold τ.

$$is^t_{\langle d_x,d_y \rangle} = \{p_1, p_2, \dots, p_n\}$$

In our representation, we preserve the direction of an *interaction_sequence*, meaning that $is^t_{\langle d_x,d_y \rangle} \neq is^t_{\langle d_y,d_x \rangle}$

Finally, we can define the following:

Definition 3.1. Given two IoT nodes, say d_x and d_y, and their interaction_sequence, say $is^t_{\langle d_x,d_y \rangle}$, we can define a *communication_set* as the set composed of all the *interaction_sequences*, performed at any time t, between d_x and d_y.

$$C_{\langle d_x,d_y \rangle} = \{is^t_{\langle d_x,d_y \rangle}\ \forall\ t\}$$

The fingerprint model. A behavioral fingerprint models how the conduct of a device d_y is perceived by another device d_x. Specifically, we introduce the following definition:

Definition 3.2. Given a pair of IoT nodes, say d_x and d_y, and their *communication_set*, say $C_{\langle d_x,d_y \rangle}$, a behavioral fingerprint is a machine learning model **ML** built by using the *communication_set* between d_x and d_y and aiming at fitting the typical conduct of d_y:

$$f_{d_x,d_y} = \mathbf{ML}(C_{\langle d_x,d_y \rangle})$$

The rationale beyond this definition is that by learning the "original" interaction style (i.e., the way a node sends packets to another one, as described by a suitable set of features, in the absence of anomalies), it could be possible to detect any anomalous behavior by analyzing possible variations in such a style.

3.2.1. The blockchain

It is the shared ledger used to record information about trust relationships among the IoT nodes. Our solution leverages a combination of behavioral fingerprinting and a community-oriented secure delegation strategy to enable safe interaction in IoT. For this reason, we will introduce a trust model (see Section 3.4) based on a consensus mechanism, to estimate the reliability of a node. In this solution, the Blockchain is used to securely store all the information needed to build the consensus mechanism and to trace the evolution of trust scores among nodes.

Although it is orthogonal to our approach, several proposals exist to create Blockchain solutions for IoT [42–45]; the only requirement in our solution is the explicit support to smart contracts.

Table 3
Example of symbol mapping.

Source port type	Packet length	TCP flag	Protocol type	IAT	Payload value	Payload value shift	Symbol
2	4	2	6	0	–	–	0
2	0	16	6	0	–	–	1
2	6	24	6	0	0	1	2
2	0	17	6	0	–	–	3
2	5	0	17	0	–	–	4

3.3. Distributed behavioral fingerprinting

This section is devoted to the description of the adopted strategy for behavioral fingerprinting. Our solution starts from the results described in [19] and extends the proposed strategy by improving the underlying deep learning model making it lighter, also through a tiny machine learning approach [46], and more suitable for an IoT scenario. The approach described in [19] considers both network parameters, as done also in [21,47,48], and introduces important new features related to the packet payload. The analysis of payload-based features is fundamental to making the behavioral fingerprint model robust also against attacks directly targeted at the surrounding Cyber–Physical System. In this case, the objective of the attacker is to leave the general interaction behavior of the victim object unaltered and to modify only the content of the exchanged control packages to force the other entities in the system to adopt specific countermeasures [49].

In particular, the approach of [19] starts from the results of [21] and considers the same set of networking-related features directly extracted from interaction packets among IoT devices. Moreover, it adds two important features related to the packets' payload. After that, an *interaction_sequence*, say $is^t_{\langle d_x, d_y \rangle}$, is converted into a corresponding sequence of symbols, say $ss^t_{\langle d_x, d_y \rangle} = \{s_1, s_2, \dots, s_n\}$, obtained on the basis of the value of the aforementioned features for each packet. The considered features list is as follows.

- **Source Port Type**. The possible values are user, system, or dynamic. This feature can be converted by mapping its values to the numbers 0, 1, and 2.
- **TCP Flags**. For this feature, the original numerical values for the considered packet are preserved.
- **Encapsulated protocol types**. Also, in this case, we can use the original numerical values already available in each packet.
- **Interval Arrival Time (IAT)**. This feature represents the time elapsed between two consecutive packets in an *interaction_sequence*. Therefore, given an *interaction_sequence*, we applied a binning transformation based on the distribution of the IAT values of the involved packets and we considered 3 indexed equal-width quantiles. In this way, each IAT value is converted into the index of the corresponding quantile.
- **Packet Length**. The engineering of this feature considers the length of all the packets involved in an *interaction_sequence* and computes the corresponding frequency distribution. At this point, the first 9 most frequent values can be mapped into 9 bins and all the other (less frequent) values can be mapped to a single final bin.
- **Payload Value**. This feature depends on the specific type of payload included in a packet. In particular, two macro-categories of payload can be considered, namely: categorical, and numerical. As for the former, the categorical payload values can be mapped to a corresponding number ranging in the interval $[0, n]$, where n is the number of the possible distinct categorical values for the payload. Concerning the latter, a binning-based strategy can be applied. In particular, continuous payload values can be mapped to 3 bins. The bins are identified based on the traffic generated during a controlled "safe" period (see Section 4 for the details). Specifically, all the payload values produced in this "safe" period can be mapped to a central bin. At this point, all the values lower than the lower bound of such a central bin will be assigned to the first bin, and all the values higher than the upper bound of the central bin will be assigned to the last bin.
- **Payload Value Shift**. This feature encodes the information related to the "variation" in the payload values for consecutive packets. In particular, it is equal to the absolute difference between the current payload value for a packet and the payload value of the preceding packet in an *interaction_sequence*.

A symbol, say s_i, is univocally associated with a combination of feature values. Packets with the same values of the involved features will be associated with the same symbol. An example of this mapping is reported in Table 3.

At this point, a behavioral fingerprint solution can be seen as a machine learning model trained to predict the next possible and admissible symbol in an *interaction_sequence*. The ratio underlying this definition is that practically speaking, learning the behavior of an IoT node implies being able to decide in advance the next most probable action that it will carry out.

For our solution, we started from the results described in [21] and in [19], in which the features based on the payload are tested. In both these works, the behavioral fingerprint model has been built as a Gated Recurrent Unit (GRU, hereafter) neural network composed of 3 neurons and a dense output layer. The size of the output layer is tailored on a specific *communication_set*, i.e., all the *interaction_sequences* between two IoT nodes; indeed it depends on the actual number of distinct symbols present in the overall *communication_set*. The experiments performed in [21] proved that this approach can reach very satisfactory results by considering a sequence of 20 symbols to be able to predict the next one. Similarly, the approach of [19], in spite of the addition of payload-related features, obtained important results using again a window of 20 symbols with a light training (less than 10 epochs with a training set of about 5K samples) to obtain an accuracy higher than 80%. Both the above approaches assume the presence of a “safe” period in which no IoT device is corrupted. This is a fundamental requirement to train the behavior fingerprint model.

In our scenario, we are considering a more general IoT context in which also legacy devices are available (i.e., devices with medium to low computational capability). To successfully apply such a solution to our context, it is fundamental to reduce to the minimum possible the complexity of the machine learning model in such a way as to directly involve the maximum possible number of nodes. Therefore, starting from the two solutions above, we tried to lighten the architecture as much as possible so that it could also be used by devices characterized by medium-to-low computational power. In particular, the solution described in [21] estimates the probability of the next packet; in our case, we reduced the problem to a classification task and we just focused on the prediction of the presence or absence of a packet as the next element of an $interaction\_sequence$. The presented neural network is again a GRU network in which we lighten up the model by cutting two of the three GRU layers and shortening the sequence of symbols required as input from 20 to 10. Interestingly, this design modification allows the achievement of pretty satisfactory performance with variations with respect to the state-of-the-art solutions of 1% accuracy at maximum (see Section 5 for all the details). Moreover, motivated by the recent introduction of *Tiny Machine Learning* approaches [50], we proceeded by converting our model into a tiny one using the TensorFlow Lite library [51] passing from a model requiring 1.6 MB to be stored to a model requiring only 415 KB. All the experiments devoted to proving the quality of the obtained results, as well as the study on the execution time for different device types, are reported in Section 5.

Finally, as demonstrated again in [19,21], behavioral fingerprint models can be leveraged for anomaly detection in IoT. Indeed, given an $interaction\_sequence$ and a sliding window, said SW, of the last k consecutive packets, the anomaly detection strategy consists of the use of the fingerprint model to predict the expected packets and to compare these results with the actual content of SW. We define the misprediction rated m_r as the number of mispredictions over the total number of packets inside a window. An anomaly is reported if the number of mispredictions observed in SW is higher than a fixed threshold (typically set to 50% of the packets in the sliding window).

3.4. Distributed consensus mechanism to estimate object reliability

This section is devoted to describing our reliability model and the underlying distributed consensus mechanism. In particular, as already stated above, one of the objectives of our solution is to provide IoT devices with a strategy to evaluate whether to instantiate a new connection with another object based on its current behavior. Due to the fully distributed nature of our approach, this solution leverages an ad-hoc consensus mechanism based on the concept of word-of-mouth among devices.

Consider a scenario in which a source device d_s may want to establish a new connection with an unknown target device d_x. Our mechanism allows d_s to obtain information about the behavior of d_x from the community of nodes belonging to its neighborhood. To do so, our approach is based on the concept of *evaluation paths* that can be formally defined as follows:

Definition 3.3. Let $pth^i_{(d_s,d_x)} = \langle d_y, d_w, \dots, d_e \rangle$ be an acyclic sequence of IoT nodes that must be contacted to reach an evaluator, say d_e, which owns a behavioral fingerprinting model of d_x. Let $E_{d_x} = \{d_e | d_e \in D \wedge \exists f_{d_e,d_x}\}$ be the set of evaluators for a target node d_x. Considering that multiple paths can exist from a source to a target node, we define the set of evaluation paths as:

$$e\_paths_{(d_s,d_x)} = \{\langle d_y, d_w, \dots, d_e \rangle \mid d_x, d_w, \dots, d_e \ \in \ D \ \wedge \ d_e \in E_{d_x}\}$$

The strategy adopted by d_s to obtain information from potential evaluators of the target d_x is as follows. First, it selects a $maximum\_depth$ value, which specifies the coverage range of the network. In particular, it indicates the maximum distance of propagation, in terms of the number of network hops, of its request (e.g., if the maximum depth is 1 only the direct neighbors will be contacted, if it is equal to 2 the direct neighbors will propagate this request to their neighbors, and so forth). After this, it will send a request packet to all its neighbors specifying the desired target d_x. At this point, the receiving nodes will verify the $maximum\_depth$ value and they will decrease it by one. If this value is greater than zero, they will continue by adding their identifier to the packet and forwarding this request to their neighbors, thus iterating this step. At each iteration, if the set of receiving nodes will contain an *evaluator* a new $path$ will be created. This concept is illustrated in the example of Fig. 2.

In this example, the source node d_s asks its neighbors, nodes d_a, d_b, and d_c, information about d_x by specifying a $maximum\_depth$ of 2. During the first iteration d_a and d_b decrease the $maximum\_depth$ to 1 and propagate the packet to their neighbors (nodes d_d and d_f). Node d_b, instead, owns a behavioral fingerprint model of d_x and, therefore, performs two actions: *(i)* it replies to d_s, thus creating the path $\langle d_b \rangle$, *(ii)* it decreases the $maximum\_depth$ and propagates the packet to its neighbors (d_f). At the second iteration, nodes d_d and d_f reply to d_a, d_b, and d_c with the information about their models towards d_x, thus creating three paths, namely: $\langle d_a, d_d \rangle$, $\langle d_b, d_f \rangle$, and $\langle d_c, d_f \rangle$. At this point, $maximum\_depth$ is equal to zero and no further propagation of the original request is performed.

To inhibit any attacker from forging fake paths, all the nodes involved will add a verifiable nonce that is univocally linked to them. To do so, we identify a solution adopting a trap-door function. In particular, when joining our system each node computes a hash-chain of size q starting from a secret $seed$ through a cryptographic hash function chf. The last value of the chain, $chf^q(seed)$, is hence made publicly available to all the other nodes (through the underlying Blockchain, see all the details below). Every time a node is involved in a new path, it will add the next (in reverse order) element of this chain. Of course, the property of the hash-chaining implies that $chf(chf^{q-1}(seed)) = chf^q(seed)$ for every q, thus providing a verification of the validity of the identifier. As a final step, as will be clearer later, the value $chf^{q-1}(seed)$ will be made publicly available to all the nodes through the underlying Blockchain, once used in a path.

At this point, given a path $pth^i_{(d_s,d_x)} \in e\_paths_{(d_s,d_x)}$, the evaluator will reply with the estimation of the trust score of the target obtained through its behavioral model. In our case, a trust score can be defined as follows.

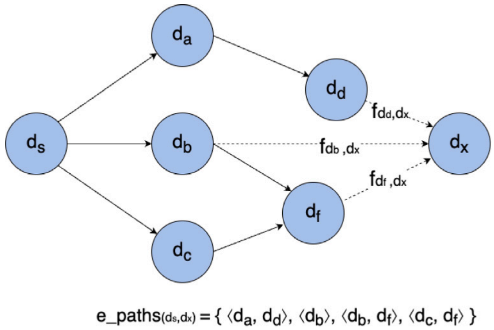

Fig. 2. An example of e_paths identification in our scenario.

Definition 3.4. Given the average misprediction rate during the most k recent *interaction_sequences*, say $\overline{m\_r}$, we define the trust score τ_{pth^i} as the complement of the average misprediction rate obtained by the model during the *interaction_sequences* between the *evaluator* and the target node:

$$\tau_{pth^i} = 1 - \overline{m\_r}$$

As shown in the example above, multiple paths can be identified from a source to a target. Each path will end with a behavioral model capable of measuring a trust score for the target. However, different conditions, such as the obsolescence of a model, a change in the state of the target, and faults in the evaluator node, could lead to wrong estimations of such a score. To reduce the impact of such anomalies, our approach adopts a consensus mechanism based on the majority of voters. In particular, we impose that, in order to properly estimate the trustworthiness of a node, at least $c+1$ paths have to return values in agreement. In our case, we consider in agreement two scores, say s_1 and s_2, such that $|s_1 - s_2| \leq tol$; where tol is a suitable tolerance threshold. It is worth observing that, as will be clearer in our Security Analysis (Section 4), given a neighbor inside the considered IoT, c is identified as the maximum theoretical number of paths that an attacker can control in such a specific neighbor. We call the set of paths returning values in agreement as *consensus set* $C\_S_{(d_s,d_x)}$. Let T_{d_s,d_x} be the average of all the trust scores of the $C\_S_{(d_s,d_x)}$, this value, also referred to as *trustworthiness* value in the following, can be used by d_s to decide whether to activate a communication towards d_x. More formally, we can give the following definition of *trustworthiness*.

Definition 3.5. Given a consensus set $C\_S_{(d_s,d_x)}$, that is a set of paths returning values in agreement, we define a trustworthiness value, say T_{d_s,d_x}, as the average of all the trust scores of the $C\_S_{(d_s,d_x)}$.

As an additional functionality, our approach estimates the quality of the contribution of each node involved in the estimation of T_{d_s,d_x}. Indeed, this average score could be used also to compute a reliability score for the participants on the different paths in $e\_paths_{(d_s,d_x)}$. In particular, the nodes involved in paths of the $C\_S_{(d_s,d_x)}$ will receive positive feedback, whereas the members of paths returning trust scores not in agreement with the majority set will receive negative feedback. The extent of the negative feedback will be directly related to the bias between the returned trust score and average score T_{d_s,d_x}. Our solution is designed so that positive feedback can balance a negative one, therefore if a node is involved in two paths, one belonging to the $C\_S_{(d_s,d_x)}$ and the other with a bias in the trust score with respect to the average, then its reliability will not have a too negative impact. The ratio underlying this choice is related to the following reasoning. A non-attacker node can be involved in paths in which there exists also an attacker and, therefore, could receive negative feedback; however, the same node will probably be involved also in paths belonging to $C\_S_{(d_s,d_x)}$. An attacker, on the other hand, will mostly be involved in paths with bias in the trust scores, possibly caused by its actions. In this way, while the balance mechanism will prevent the reliability of honest nodes from going down, this effect should not happen for attackers. At this point, we can give the following definitions:

Definition 3.6. Given a consensus set $C\_S_{(d_s,d_x)}$ and the set of paths $\overline{C\_S}_{(d_s,d_x)} = e\_paths_{(d_s,d_x)} \setminus C\_S_{(d_s,d_x)}$ with trust scores not in agreement with it, we define $\Gamma(C\_S_{(d_s,d_x)}, d_y)$ (resp. $\Gamma(\overline{C\_S}_{(d_s,d_x)}, d_y)$) as a function returning the paths of $C\_S_{(d_s,d_x)}$ (resp. $\overline{C\_S}_{(d_s,d_x)}$) involving the node d_y:

$$\Gamma(C\_S_{(d_s,d_x)}, d_y) = \{pth_i | d_y \in pth_i \wedge pth_i \in C\_S_{(d_s,d_x)}\}$$

At this point, the variation of the reliability for a node can be estimated as shown in Eq. (1).

$$\Delta\_RL_{d_y} = \sum_{pth^i \in \Gamma(\overline{C\_S}_{(d_s,d_x)}, d_y)} -|\tau_{pth^i} - T_{(d_s,d_x)}| + \gamma \cdot |\Gamma(C\_S_{(d_s,d_x)}, d_y)| \tag{1}$$

Here, γ is a parameter used to tune the impact of a positive contribution with respect to a negative one. In our scenario, all the nodes involved will start with a default reliability value, say r. Given a node d_y, its reliability value will be decreased of the value $\Delta\_RL_{d_y}$, if $\Delta\_RL_{d_y}$ is less than zero; no updates at its reliability will be done, otherwise.

Actually, as described in Section 3.2, our approach leverages a Blockchain-based solution to trace, in a fully distributed fashion, the evolution of the behavior of objects when interacting with each other. In our application scenario, we consider a managed Blockchain supporting smart contracts. Our solution is based on a smart contract, say SM deployed on the Blockchain, which gathers transactions from IoT nodes. In the solution above, once the paths have been identified and the source node has received all the trust scores, it creates a transaction towards the Blockchain reporting information about all the paths, the identifier, and the verifiable nonce of the involved nodes and the corresponding trust scores. At this point, SM will be executed to perform the following tasks:

1. verify the nonce for each involved node[1];
2. identify the *consensus set*, if it exists;
3. compute the average trust score;
4. compute Δ_RL for each node;
5. update reliability values for the nodes with $\Delta\_RL < 0$;
6. publish a transaction reporting the results of the previous steps.[2]

The information available in the Blockchain through SM is, then, used by IoT nodes to identify corrupted nodes that should not be involved in the next interactions. In particular, we assume that all the nodes with a reliability lower than a control threshold c_{th} will not be engaged in future actions.

Algorithm 1 summarizes the steps of our consensus mechanism for the object reliability assessment. Observe that *evalPaths* is a recursive function used to compute all the e_paths between the source and target node in the network.

Algorithm 1: Consensus mechanism for object reliability

```
Data: d_s, d_x, d_e ;                                  /* source, target and evaluator node */
pth^i_(d_s,d_x) ← ⟨d_y, d_w, ⋯ , d_e⟩ ;                 /* path towards d_e */
E_(d_x) ← {d_e | d_e ∈ D ∧ ∃ f_(d_e,d_x)} ;             /* set of evaluators for d_x */
e_paths_(d_s,d_x) ← {⟨d_y, d_w, ⋯ , d_e⟩ | d_x, d_w, ⋯ , d_e ∈ D ∧ d_e ∈ E_(d_x)} ;   /* evaluation paths */
Result: T_(d_s,d_x) ← d_x reliability
evalPaths(d_s, maximum_depth, d_x) is
    pth^i_(d_s,d_x) ← pth^i_(d_s,d_x) + d_i;
    maximum_depth ← maximum_depth − 1;
    if maximum_depth ← 0 ∨ d_s ∈ E_(d_x) then
        return pth^i_(d_s,d_x) ;
    else
        while d_i ∈ Neigh_s do
            d_s sends a request to d_i for d_x;
            e_paths_(d_s,d_x) ← e_paths_(d_s,d_x) + evalPaths(d_i, maximum_depth, d_x);
        end
    end
end
d_s selects maximum_depth > 0;
d_s computes e_paths_(d_s,d_x) ← evalPaths(d_s, maximum_depth, d_x);
while pth^i_(d_s,d_x) ∈ e_paths_(d_s,d_x) do
    d_s requests to d_e: τ_(pth^i);
end
d_s creates a Blockchain transaction with all the information of nodes ∈ E_(d_x);
```

3.5. Community-oriented secure delegation

As stated above, our application scenario embraces a situation in which heterogeneous devices, belonging to the three categories described in Section 3.1, collaborate to build our secure interaction scheme. To achieve this objective, we also propose a secure delegation mechanism according to which capable devices (belonging to the CD category) can delegate the training of behavioral fingerprint models to devices of the PD category. Similarly, nodes of the BD category can again delegate powerful devices (belonging to the PD category) to train their models and can leverage both CD and PD nodes for the model inference.

[1] Observe that, a node can be involved in multiple paths. In this case, it will use different values of its inverse hash chain. SM will verify the consistency of all the values for a single node by linking them to the previously published element of the chain.

[2] The transaction will also include the last verified nonce for each involved node.

In particular, our approach leverages a combination of the Blockchain solution described above and an IPFS-based [52] strategy to exchange information about training data and the obtained models. IPFS (InterPlanetary File System) is a peer-to-peer fully distributed file system that is typically exploited in combination with Blockchain technology to enable secure data exchange.

At this point, our secure delegation mechanism works as follows. Given a node d_x, the activation of a secure delegation starts by identifying, among the neighbors of d_x, reliable nodes to be delegated. The delegation can concern two aspects, namely: (1) the training of a behavioral fingerprint model; *(ii)* a trained model inference. As for the former, only nodes of PD can be involved. The latter instead can be demanded to both nodes from PD and CD. In any case, to identify reliable nodes, d_x can leverage the information from the smart contract SM defined in Section 3.4 to obtain, for each node in its neighborhood $Neigh_{d_x}(i)$ (at any level i), the corresponding reliability values RL. At this point, d_x will send a *delegation request* to all the nodes having a reliability higher than the control threshold c_{th} (see Section 3.4). Depending on their current status (including battery condition, traffic overhead, and so forth) each neighbor will decide whether to accept the request from d_x or not.

In the positive case in which at least one node, say d_n, accepted the *delegation request*, our solution will proceed with the steps described below.

First, it is worth observing that, the construction of a behavioral fingerprinting model according to our scheme requires the exploitation of sensitive information available in the data exchanged between d_x and a target node in the system, for which it wants to build the behavioral fingerprint (see Section 3.3 to have a list of the information pieces required by our solution). For this reason, a crucial aspect to consider when enabling a delegation strategy is the privacy concern. Hence, our secure delegation scheme includes a privacy-preserving strategy so that d_x can share the required information with d_n, still preserving the privacy of the original communication between it and the target node. To do so, as described below, we leverage the peculiarity of our behavioral fingerprint model to work with a sequence of symbols. As seen in Section 3.3, each symbol corresponds to a specific combination of the features extracted from the packets of an $interaction\_sequence$. Therefore, to preserve the privacy of the communication between d_x and the target node, our secure delegation strategy enforces that such a sequence of symbols is converted into a corresponding sequence of values using a cryptographic hash function for the mapping.

Secure delegation: training a behavioral fingerprint model. The node d_x collects the training data of its target and derives the corresponding symbols using the feature engineering task described in Section 3.3. To make our solution robust to privacy issues related to the knowledge included in the symbols, we impose that d_x applies a salt-based cryptographic hash function to each symbol, $hs_i = chf(s_i, salt)$. The choice of the adopted cryptographic hash function depends on the trade-off between the need for privacy protection and the computational effort to obtain the hashed symbols. The hashed dataset will then be uploaded into a folder of IPFS and the reference of the address of such a folder will be sent to d_n. At this point, d_n will proceed by training a behavioral fingerprint model using the data from d_x.[3] Finally, if the secure delegation concerns only the training phase, d_n will upload the trained model to IPFS and will share the position with d_x. Instead, if the delegation concerns also the model inference, d_n could retain the trained model in its memory to support d_x during the subsequent model inference phase.

Secure delegation: model inference. Given a trained model, the node d_x will gather the data (set of symbols) related to a sliding window, according to the approach described in Section 3.3. For each symbol s_i of this sliding window, d_x will compute the corresponding hashed version using the salt-based cryptographic hash function $hs_i = chf(s_i, salt)$. The obtained values will be used as input to the trained model. At this point, two situations may occur: either d_x can directly perform a model inference or d_x will share this data with its delegate d_n. In both cases, the output will be the misprediction rate measured in the corresponding sliding window.

Fig. 3 sketches the solution described above and the use of IPFS to exchange both the training data and the trained behavioral fingerprint model.

Algorithm 2 summarizes the steps of our community-oriented secure delegation mechanism.

4. Security model

This section is devoted to the security model underlying our solution. In the next sections, we introduce both the attack model and the security analysis proving that our approach works also in the presence of attacks. Since the main contribution of our strategy is the construction of a mechanism to evaluate the trustworthiness of IoT nodes, in our security analysis we consider classical attacks on reputation systems such as those mentioned in [2,53].

4.1. Attack model

Preliminary, observe that our solution in the stationary scenario considers enough nodes available to carry out the steps required by our approach. As a consequence, we do not focus on the initial stages possibly characterized by anomalous situations in which the IoT network is not yet active or complete.

With that said, before analyzing the security properties of our model, we outline the following starting assumptions:

---

[3] Observe that, again for privacy reasons, the identifier of the target node of the model is not available to d_n.

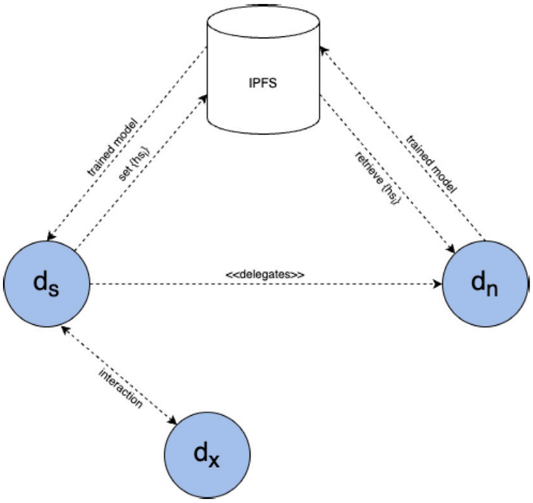

Fig. 3. An example of our secure delegation strategy.

Algorithm 2: Secure Delegation mechanism

Data: d_x, d_t, c_{th} ; /* source node, target node, control threshold */
$hs_i \leftarrow chf(s_i, salt)$; /* salt-based cryptographic hash function for symbol s_i */
Result: $\Delta\_RL_{d_t}$; /* d_t misprediction rate */
d_x gets from the Blockchain the list of nodes $Neigh_{d_x}(i)$;
while $d_n \in Neigh_{d_x}(i)$ **do**
    if $RL_{d_n} > c_{th}$ **then**
        d_x sends a delegation request to d_n;
        if d_n *status* ← *ok* **then**
            d_n accepts d_x request;
            d_x collects training data of d_t;
            d_x applies $hs_i \leftarrow chf(s_i, salt)$;
            if *secure delegation includes only the training phase* **then**
                d_n trains a behavioral fingerprint model using d_x data;
                d_n uploads the trained model to IPFS;
                d_n shares the position with d_x;
            else
                d_n (or d_x) computes $\Delta\_RL_{d_t}$;
            end
        end
    end
end

A.1 An attacker can control at most c paths of a set $e\_paths_{d_s,d_x}$, between any pair of nodes, d_s and d_x.[4]
A.2 There exists a safe stage in which behavioral fingerprint models can be computed in the absence of attacks.
A.3 An attacker cannot control all the behavioral fingerprint models associated with IoT nodes.
A.4 An attacker has no additional knowledge derived from any direct physical access to IoT objects (especially about the paths linking nodes).
A.5 The Blockchain technology exploited to implement the support public ledger is compliant with the standard security requirements already adopted for common Blockchain applications.
A.6 An attacker cannot have access to the secrets to generate the hash chains of IoT nodes.
A.7 The adopted cryptographic hash function is robust against collision, preimage, and second preimage attacks.

[4] Observe that, this is a common assumption for distributed domain scenarios, in which the majority of users or nodes in a network or a system can be considered honest at any time [54–56].

As stated above, our model ensures a list of security properties (SP, in the following), which are listed below.

SP.1 Resistance to attacks to the Blockchain and the smart contract technology.
SP.2 Resistance to Self-promoting Attacks.
SP.3 Resistance to Whitewashing or Self-serving Attacks.
SP.4 Resistance to Slandering or Bad-mouthing Attacks.
SP.5 Resistance to Opportunistic Service Attacks.
SP.6 Resistance to Ballot Stuffing Attacks.
SP.7 Resistance to Denial of Service (DoS) Attacks.

4.2. Security analysis

This section is devoted to the analysis of the security properties presented above to prove that our approach can ensure them. In the next sub-sections, we analyze each of these properties in detail.

4.2.1. SP.1 — Resistance to attacks to the blockchain and the smart contract technology

This category of attacks aims at finding vulnerabilities in the Blockchain and the smart contract technology adopted in our approach. This technology has been the subject of studies from the research community in recent years. Anyway, the security of Blockchain is still under the spotlight and represents an open issue [57–60]. The approach presented in this paper is not devoted to facing security challenges on Blockchain but focuses on its application as a secure public ledger to support a collaborative fully distributed approach for secure device interactions in IoT. Therefore, we assume that the Blockchain and the smart contract technology can be considered secure (Assumption A.5). Our approach is, hence, orthogonal to any solution, available in the scientific literature, to improve the security of Blockchain that can, hence, be applied in our context to guarantee our assumption.

4.2.2. SP.2 — Resistance to self-promoting attacks

In this case, an attacker controlling a node or a set of nodes could work to manipulate the reliability and trustworthiness of a target node. This attack can be done by either a single node or through joint actions of different colluding nodes. Concerning the first case, actually, a node cannot alter the perception that other nodes have of it, because the trustworthiness of a node is estimated using behavioral fingerprinting. Therefore, any variation in its typical behavior would increase the misprediction of the models associated with this node. To alter this value, the attacker should have access to the nodes holding a fingerprint model towards the target. However, this cannot happen thanks to Assumptions A.2, A.3, and A.4. Also, the reliability cannot be altered because it is associated with the collaborative approach described in Section 3.4. As such, the attacker should include its node always in paths belonging to *consensus sets* to avoid the detriment of its reliability. However, for Assumption A.4 an attacker cannot know the topology of the network nor force the inclusion of a node into a path due to the security mechanism leveraging the cryptographic hash chaining strategy described in Section 3.4, which thanks to Assumptions A.6 and A.7 cannot be broken or forged.

According to the second strategy, an attacker could leverage different nodes colluding to positively change the trustworthiness and reliability of a node. As for the trustworthiness, once again, due to Assumptions A.2, A.3, and A.4 the attacker cannot control all the behavioral fingerprint models; moreover, she/he cannot forge a trustworthiness score for the target node by leveraging a collaborative self-promoting attack. Indeed, thanks to Assumption A.1, the attacker can only control c paths in the set of e_paths between the target node and any other node in the system. However, the trustworthiness value of a node can only be estimated if there exist at least $c+1$ paths with a trust score in agreement. Therefore, also thanks to Assumptions A.6 and A.7, this attack cannot happen. As for the reliability, as already seen for the single node version of this attack, because the estimation of the reliability derives from the consensus mechanism and since the attacker cannot forge paths nor control more than c real paths in an e_paths set (Assumptions A.6, A.7, and A.1), this attack cannot be carried out.

Finally, in any case, the attacker cannot alter the computed trustworthiness and reliability scores from the Blockchain thanks to Assumption A.5 and the security property SP.1.

4.2.3. SP.3 — Resistance to whitewashing or self-serving attacks

This attack concerns any attempt of a malicious node to clean its trustworthiness and reliability scores to be involved again in the activity of the network. In our approach, trustworthiness scores are computed through the evaluation of the behavior of the target nodes by exploiting an existing trained fingerprint model. Therefore, also thanks to Assumption A.3, an attacked node cannot whitewash this score as it depends on the models owned by the surrounding nodes. As for the reliability of a node when participating in the collaborative consensus-based mechanism, the corresponding value is permanently stored in the underlying Blockchain. In the absence of attacks, the reliability is set initially to a default positive (over the control threshold) value r, and it can only be reduced (no positive increment) if the interactions of the node during the execution of the consensus algorithm are not evaluated positively by the community. If the reliability of a node is under a control threshold (see Section 3.4), this node will not be involved in the next activities by the other members of the IoT network. Depending on the security requirement of the considered scenario, an under-threshold reliability score can stay so for a time interval of ban, say t_{ban}. After that, the reliability is restored to the default initial value r. Of course, for critical scenarios, this value could be infinite so that the restoration of a node could be done only through the manual intervention of a system administrator.

It is worth observing that, in IoT, one of the main problems is related to the difficulty of assessing a unique identifier for a device. In our case, there is a direct relationship between a device identifier and its profile on the underlying Blockchain. Of course, an attacker could perform a whitewashing attack on a node by exiting the system and re-introducing the device with a different (forged) identifier. To face this situation, we adopt a pessimistic attitude approach, which imposes that newly introduced devices will be associated with a negative (under the control threshold) reliability [61–63]. Under this assumption, a new device will start in a banned state (no other node will interact with it) until its reliability is set to the default value after t_{ban}. In this way, attempting a whitewashing by forging a new identifier for a device would result again in the node being banned for t_{ban}. Therefore, no advantage is obtained by the attacker.

4.2.4. **SP.4** — *Resistance to slandering or bad-mouthing attacks*

This attack occurs when an intruder tries to distort innocent nodes' reputations by sending negative reputation values about them. In this way, the attacker ruins the trustworthiness and/or the reliability of a target node to force its exclusion from the system. Classically, there are a number of strategies used as security countermeasures against this kind of attack that generally consider: *(i)* nodes' historical trust value; *(ii)* nodes' current trust value and *(ii)* path trust value [64]. In our approach we partially leverage all of these countermeasures exploiting and combining them with the additional potentialities of our solution.

In particular, as for trustworthiness, this can only be estimated through existing trained behavioral fingerprint models. For Assumption **A.3**, the attacker cannot control all the models referring to its target. Our approach is designed to use a consensus strategy to estimate the trustworthiness of a node by leveraging all the information obtained by the possibly different trained models describing its behavior. When it comes to reliability, instead, this is evaluated based on the quality of the interactions of a node during the execution of the consensus algorithm. In this case, the adversary can attempt to execute two different strategies. Because the overall reliability variation derived from an interaction depends on the difference between the result provided by the *consensus set* and that of the paths in which a node is involved, the attacker could force a wrong estimation for a path it is also involved in. In this way, all the nodes (including the attacker) in such a path will be negatively evaluated. Our strategy implies a balancing mechanism according to which if a node is involved in both a positive and a negative path, no negative variation is recorded for its reliability. Our assumption here is that a non-malicious node will be involved in different paths that cannot be all controlled by an attacker (Assumption **A.1**). Of course, if a node is only linked to an attacked one, it can be considered under the control of the attacker itself, and, hence, it should be excluded by the system. According to the second strategy, instead, the attacker could try to forge false paths returning results very different from any *consensus set* and involving the target node. In this way, she/he could effectively cause a detriment on the reliability of such a node. As a countermeasure, our approach includes a mechanism to ensure the real membership of a node to a path. Indeed, when a path is formed, each involved node would add a verifiable nonce uniquely related to it. This solution is obtained through a cryptographic hash chain that each node builds when joining the system. The first element of the (inverse) chain is stored on the Blockchain and is used to verify the correctness of the following values. At the end of the execution of the consensus algorithm, the used elements of the chain are made publicly available on the Blockchain (see Section 3.4). Of course, this implies that no one could re-use the information of a path to perform a *reply-attack* because the verifiable nonce must not be already available in the Blockchain (so that $chf(chf^{q-1}(seed)) = chf^{q}(seed)$). This strategy, also thanks to Assumptions **A.5**, **A.6**, and **A.7** makes our approach robust against this attack.

4.2.5. **SP.5** — *Resistance to opportunistic service attacks*

A malicious node could selectively behave good or bad, opportunistically. This strategy can be carried out on both the standard interactions with the other nodes and on the interactions related to the consensus mechanism. As for the former aspect, behavioral fingerprint models can successfully detect any change in the behavior even if selective. Indeed, these models are built under Assumption **A.2** in which no malicious behavior is present. As for the second strategy, the idea underlying it is that the attacker knows of the balancing effect in the computation of the reliability and tries to leverage this feature to partially attack the network while still preserving its status. However, because our approach enforces the existence of a *consensus set* to evaluate the trustworthiness of a node, thanks to Assumption **A.1**, no advantage can be obtained by the attacker. Of course, she/he could selectively cause the detriment of the reliability of nodes that are only linked to the attacker (and, hence, are not involved in other *honest* paths). However, as stated above, if a node is reachable only through the attacker, it can be assumed under her/his control. For this reason, the consequent isolation of such a node (due to low reliability) is actually intended in our solution.

4.2.6. **SP.6** — *Resistance to ballot stuffing attacks*

According to this typology of attack, an adversary could exploit the position of a node in the network to positively increment the trustworthiness and/or reliability of a target (malicious) node. As for trustworthiness, the attacker would need to control all the behavioral fingerprint models of the target, which is forbidden by Assumption **A.3**. Moreover, she/he cannot control all the paths towards the target so to selectively hide the *honest* models and privilege the controlled ones to obtain a *consensus set* (Assumption **A.1**). Finally, for Assumptions **A.6**, and **A.7**, she/he cannot forge artificial paths to favor the access to controlled (malicious) models. As for reliability, the attacker cannot positively increment it for a node because our approach records only negative variations. Once under the control threshold, the reliability can be restored only after a ban period t_{ban} (see the description for the security property **SP.3**). Finally, recall that reliability scores are stored on the Blockchain, which enforces that no devices can corrupt or change such scores, either positively or negatively.

4.2.7. SP.7 — Resistance to Denial of Service (DoS) attacks

Denial of Services (DoS) attacks try to bog down a system by overflooding it with a very high number of (dummy) transactions. In our approach, this attack could also result in the impossibility of nodes to gather information about the trustworthiness and reliability of the other nodes. Indeed, the goal of the attacker could be to prevent the identification of the *consensus set* to estimate the trustworthiness of a node. Although it may represent a problem in our context, our strategy does not directly deal with this typology of attack. However, it is important to mention that, our approach does not add any advantage to an adversary performing this typology of attack. For this reason, any existing strategy conceived to prevent/face DoS attacks in IoT could be included in our approach, such as the solutions presented in [65–68].

In IoT, DoS attacks can take the form of a Sleep Deprivation Attack (SDA, hereafter) whose objective is the power consumption of the devices to exclude them from the system through a battery drain. As for this aspect, our approach natively supports a countermeasure. Indeed, when performing a DoS attack a node alters its standard behavior. Such information is detectable by our behavioral fingerprint models; therefore, IoT nodes can safely discard all the requests from nodes whose behavior is anomalous, thus preventing SDA attacks from happening.

5. Experiments

This section is devoted to the experiments for validating our approach. In particular, in the next sub-sections we report in detail the performance evaluation of our solution to build a behavioral fingerprinting model, the tests to identify the best tuning configurations, the experiments devoted to assessing the quality of our delegation strategy, as well as the performance of the overall approach on different type of devices, and, finally, we show the results of our solution for the anomaly detection using our consensus-based algorithm.

5.1. The underlying dataset

To conduct our experiments we started from the same dataset and approach of [19]. In particular, we leveraged the dataset described in [69] and available at https://iotanalytics.unsw.edu.au/attack-data. The dataset is composed of two parts: the raw packet traces, and the flow counters. The data concerns the interactions of 27 IoT nodes; among these 10 devices were also included in attack traffic. Benign and attack traffic has been recorded for two periods, namely from May 28th 2018 to June 20th 2018, and from October 10th 2018 to October 29th 2018. The attack traffic was properly labeled and occurred in the periods from June 1st 2018 to June 8th 2018, on June 20th 2018, and from October 20th 2018 to October 27th 2018. The information about the attacks comprises the start and end time, the flow influenced by the attack, the type of the attack, the bit-rate of the attack, the attacker identifier, and the victim identifier.

A limitation of this dataset in our context is related to the fact that the considered scenario concerns a centralized environment in which a *hub* collected the message exchanges. Because the central *hub* is not the intended recipient of such messages, the collected packet information cannot include payload data (due to the message encryption). However, payload-based features are an important component in our approach and, therefore, we adopted the strategy proposed in [19] to alter the previous dataset and include synthetic payload data. In particular, among all the packets available in the original dataset, the ones carrying a payload can be identified by checking the PSH TCP flag. At this point, we used the algorithms originally proposed in [19] to generate both quantitative and categorical payload data. Quantitative payload generation simulates devices like temperature, pressure, or humidity sensors (Algorithm 3).

Algorithm 3: Algorithm for quantitative payload generation [19]

Data: R = [lower bound, upper bound], HOP, n
Result: PL; /* PL is a list of n payload values */
$i \leftarrow 1$;
$PL\_0 \sim U(R)$;
while $i < n$ **do**
  $PL_i \sim \mathrm{U}(\mathrm{R} \cap [PL_{i-1} - HOP, PL_{i-1} + HOP])$;
  $i \leftarrow i + 1$;

In practice, this algorithm takes in input the range values in which the generated payloads should be contained (R), the number of consecutive payload values that should be generated (n), and the maximum gap admitted between 2 consecutive payload values ($HOP \geq 0$). Hence, it generates the first payload value PL_0 through a uniform sampling in R. Each subsequent payload value, say PL_i, is uniformly sampled in an interval centered around the value of the previous payload, say PL_{i-1}, with a size of $2 \cdot HOP$. The algorithm controls the rate of variation of the generated quantitative payload values based on the HOP parameter.

As for the categorical payload generation, we adopted the Algorithm 4 originally proposed, once again, in [19].

This algorithm accepts the list of categorical values (R), the total number of different payloads to be generated (n), and a stability period representing the time interval in which the categorical value should not change. At this point, the algorithm extracts a duration for a categorical value in the range defined by the stability period. After that, it randomly extracts the corresponding value from R. This value will be contained in all the packets exchanged during a time window equal to the duration extracted above. The stability period is used to control the oscillation frequency of the categorical payload. Some statistics about the obtained dataset are reported in Table 4.

Algorithm 4: Algorithm for categorical payload generation [19]

Data: $R = \{val_1, \dots, val_q\}$, n, $stabilityPeriod$=[min, max]
Result: PL; /* PL is a list of n payload values */
$i \leftarrow 0$;
$PL_0 \sim U(R)$;
while $i < n$ **do**
  $STAB \leftarrow stabilityPeriod$;
  if $i + STAB < n$ **then**
    $val \sim U(R)$;
    $PL_{i,\dots,i+STAB} \leftarrow val$;
    $i \leftarrow i + STAB + 1$;
  else
    $val \sim U(R)$;
    $PL_{i,\dots,n} \leftarrow val$;
    $i \leftarrow n$;

Table 4
Statistics of the dataset considered in our study.

Type of communication	Min # of packets	Max # of packets
Benign	12,793	97,256
Benign with payload	4,670	39,000
Malign	6,971	89,148
Malign with payload	2,196	8,694

Table 5
Accuracy of the models on Test Set.

Model	Accuracy	F-Measure
Model of [19]	79%	77%
Our Model	78.6%	76.6%
Tiny Machine Learning Version	78.6%	76.6%

5.2. Analyzing the performance of our lightweight fingerprint model

This section reports the details about the training and the performance obtained for our lightweight behavioral fingerprinting model. To conduct this experiment, we randomly selected the *communication set* of three different devices from the dataset introduced in Section 5.1. As stated in Section 3.3, our solution starts from the results reported in [19,21], and we strove to build a lighter version of the models proposed in this previous study to cope with the limitation of the considered IoT context. To prove the performance of our solution, we carried out a comparative evaluation between our new model and the model of [19]. The results of this comparative analysis are reported in Table 5.

The task addressed is the same for each of the models analyzed in this table, i.e., the prediction of the next symbol of an input sequence. Our objective is to obtain a lighter model than the one used in [19] to guarantee the suitability of our solution to the widest possible range of smart objects in modern IoT scenarios. For this reason, as explained in Section 3.3, we reformulated the prediction problem into a classification problem in which, instead of estimating the probability of the next symbol, we limited it to the prediction of the presence or absence of a symbol as next element of a sequence. By doing so, we were able to reduce the complexity of the adopted deep learning model. From the analysis of Table 5, we can see that the performance of our simplified model mostly matches that of [19] with a difference of less than 1% in both accuracy and f-measure. The already negligible performance difference is even less impacting if we consider that the model is then used in a window-based mechanism for anomaly detection. The dimension of the windows is designed to estimate the average behavior of a node considering different packet exchanges during a monitoring period. Such a window-based approach is intended to smooth out any prediction error caused by the considered model. Further experiments on the role of the window-based mechanism, confirming the suitability of our lighter model for anomaly detection, are reported in Section 5.3. The consideration above remains true also for the tiny version of our model obtained through the conversion done with TensorFlow Lite [51]; the quantization applied by this strategy does not impact the model performance which preserves the accuracy results.

It is worth mentioning that the lightning process has allowed us to reduce the number of model parameters by 60% as shown in Table 6, while maintaining the desired performance.

This complexity reduction of the proposed model allowed us to obtain pretty satisfactory performance on several hardware configurations as will be shown in the next sections. This aspect, along with the previous considerations on the satisfactory accuracy results with respect to previous approaches, makes the tiny machine learning version of our reduced model the optimal solution to be adopted in the reference IoT context.

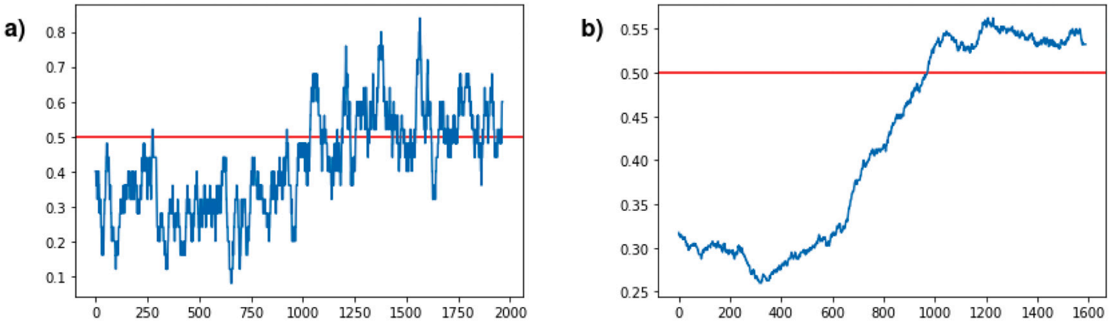

Fig. 4. Traffic analysis with windows of different sizes: *(a)* 25 packets size and *(b)* 400 packets size.

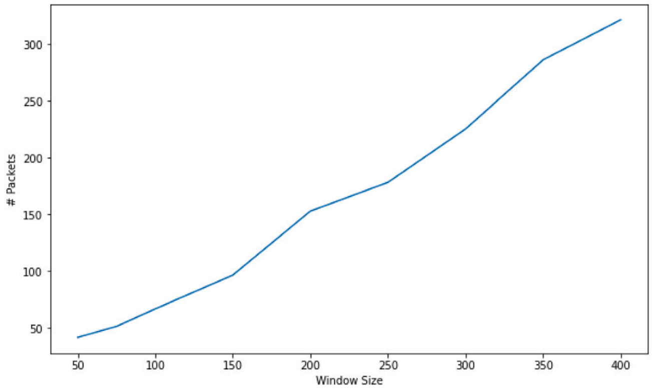

Fig. 5. Number of packets required to detect an anomaly.

Table 6
Number of parameters.

Model	Number of parameters
Model of [19]	260 k
Our model	106 k

5.3. Anomaly detection: Window size selection

After training the model for the prediction of the next symbol, it is possible to build our solution for anomaly detection to identify changes from the normal behavior of the analyzed object.

To do so, we check the misprediction rate of the next symbol in a given window of consecutive packets. In our approach, we detect an anomaly when more than half of the packets predicted are different from the packets received. Following the reasoning above, our anomaly detection strategy strongly depends on the correct size of the chosen window. Therefore, using the models obtained from the previous experiment, we proceeded by analyzing the misprediction rate of the devices using windows of different sizes. In particular, in our experiment, we considered an *interaction_sequence* of 2,000 packets, in which the first half part of the sequence represents benign traffic and the second half malign one. We tested our solution with different window sizes and, for each of them, we analyzed the misprediction rate and, in particular, we focused on the difference between the maximum and minimum peaks of this curve. As a result, we obtained that the bigger the window size the more stable the obtained curve. In Fig. 4, we report the result of this analysis for both a window size of 25 packets and one of 400 packets. In this figure, the x-axis reports the sliding window number, while the y-axis indicates the misprediction rate. The anomaly threshold is fixed at 0.5 (meaning that an anomaly is detected if half of the packets inside a window are incorrectly predicted).

As we can see from this figure, the size of the window drastically changes the oscillation of the misprediction rate curve. This oscillatory behavior can, of course, lead to a not unstable anomaly detection.

Intuitively, bigger window sizes would result in a greater number of packets to detect an anomaly. In particular, from Fig. 5, we can see how the number of required packets to detect anomalous behavior is directly proportional to the size of the window.

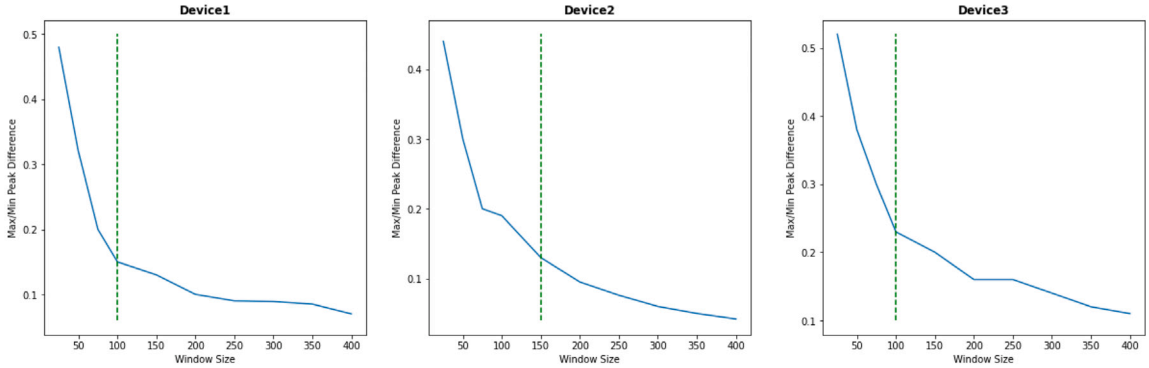

Fig. 6. Difference between Minimum and Maximum Peaks.

Table 7
Training and model inference execution times for different classes of device.

	Single-Core ARM	Raspberry Pi 4	Powerful device
Training Epoch Time (seconds)	671	30	3
Prediction Time (milliseconds)	1500	86	7
TinyML Prediction model Time (milliseconds)	0.4	0.4	0.004

Therefore, to select the best window size, starting from the difference between the maximum and minimum peaks of the misprediction curve, we leverage the *Kneedle* algorithm [70]. Specifically, this algorithm tries to find the elbow/knee in a curve by selecting the *right* operating point for a given system.

In Fig. 6, we demonstrate the application of this strategy to identify the best window size for *interaction_sequences* involving three different devices. The results show that the *Kneedle* algorithm returned a correct window size of about 100 packets for all three devices.

5.4. Analysis of the execution time for different types of device

In this section, we study the execution time required by the different tiers of devices to train and execute the fingerprinting model described in Section 5.2. The objective of this experiment is to verify the advantages, in terms of running time, introduced by our secure delegation strategy.

In this experiment, we compare three different devices, one belonging to the class of *Basic Devices*, one to the class of *Capable Devices*, and one belonging to the *Powerful Device* class. In particular, as for the *Basic Device*, we used an emulation device equipped with a single core 1GHz Arm processor (ARM 1176) using the QEMU emulator. This kind of processor is usually employed in objects such as smart locks, smart thermostats, or simple boards like the first version of the Raspberry Pi family. Concerning the *Capable Device* class, we selected a general-purpose single-board personal computer, namely *Raspberry Pi* 4. These devices are very common examples of IoT nodes [71,72]. Finally, as for the *Powerful Device* class, we employed a device equipped with an eight-core Desktop CPU (Ryzen 7 5800x).

We tested the performance of these devices for both training and model inference and we reported the results in Table 7.

From this table, we can see how the class of *Basic Devices* requires more than 10 minutes to complete an epoch compared to the 3 seconds required by the class *Powerful device*. The *Capable Device* class would require about 30 seconds for each training epoch. In our experiment, we found that the training needs about 10 epochs to reach satisfactory performance with a training set of about $5K$ samples. For this reason, the overall training would be mostly prohibitive for the *Basic Device* class. Instead, it would require moderate energy consumption for the class of *Capable Devices* (about 5 minutes of computation, on average). The training cost would be negligible for the *Powerful Device* class (about 30 seconds on average). Thus making our secure delegation strategy particularly advantageous in this case.

As for the model inference, the time required to predict a single symbol appears sustainable in all the cases, with a maximum of 1.5 s for the *Basic Device* class. Interestingly, this value reduces drastically after the conversion of our model to a TinyML solution.

5.5. Anomaly detection using the consensus algorithm

The experiments described in this section aim at evaluating the performance of our strategy for distributed anomaly detection. In general, in our scenario, an anomaly in the behavior of a node can be caused by either a hardware malfunction or an undergoing cyber attack. Here, we focused only on anomalies caused by attacks and we studied the capability of our solution to detect them. We considered two categories of active attacks that a malicious IoT node can perform, namely: *(i)* direct interaction attacks; *(ii)* attacks on the consensus mechanism. To the former category belong attacks that typically alter the communication behavior of controlled nodes to cause damage to the surrounding environment, such as Denial of Service, Sleep Deprivation, Ping of Death, ARP Spoofing, TCP SYN-ACK reflection, and so forth. The latter category, instead, comprises attacks targeting the reputation model and, hence, our consensus mechanism. Examples of attacks in this category are Self-promoting, Slandering, and Bad-mouthing.

We start our evaluation by considering the former category of attacks above. Such attacks cause a variation in the communication behavior of malicious nodes and, therefore, to validate our proposal we tested the performance of our distributed behavioral fingerprint-based solution to detect such behavioral anomalies. Indeed, although, thanks to the extreme lightness of our model, all the classes of devices can train a behavioral fingerprinting model, either directly (Powerful Devices and Capable Devices) or through our secure delegation mechanism, and perform model inference, this can happen only in presence of a *safe* period that can be assumed, for example, during the *start-up* of the network. For this reason, only a percentage of nodes can ultimately have behavioral models for some of their neighbors. To enable the propagation of trustworthiness data on the whole IoT network, we designed a distributed consensus mechanism to estimate the trust values associated with a node (see Section 3.4). Therefore, in this experiment, we tested the average time required by a generic node in the network to detect anomalous behavior of a peer, estimated through our distributed consensus mechanism. It is worth explaining that, in this first experiment, we did not consider the possible coexistence of attacks of the second category above. To do so, we built a simulated IoT by using device emulation solutions (such as QEMU and VirtualBox) equipped with a telemetry system (to log all packet exchanges) written in Python. We implemented our solution and used the dataset described in Section 5.1 to simulate real interactions among the nodes. To perform our experiment, we injected data from a malicious node in the simulated network and collected all the information and data exchanged among the nodes. In particular, starting from the dataset generated in Section 5.1, we crafted different communications between devices inside the network maintaining the integrity of the normal behavior for the involved nodes. The crafted communications are composed of half-benign and half-malign traffic in order to monitor the shift between normal behavior and anomalous one. Specifically, we measured the number of malicious packets necessary to detect an anomaly through our consensus algorithm. We found that this number is about 61 on average. Clearly, depending on the dynamicity of the network (in terms of the frequency of packet exchange among nodes) this could correspond to very different detection times, spanning from a few seconds to minutes. However, it is worth noting that, when a node is compromised and exhibits an anomalous behavior, direct neighbors using a behavioral fingerprinting model will detect the anomaly almost instantaneously. The additional 61 packets reported above represent the average effort required by our system so that *any* interested node inside the whole IoT network can detect a behavior change of a malicious actor.

At this point, we proceeded by assessing the capability of our solution to contrast the second category of attacks, i.e., those targeting our underlying consensus-based reputation model. Here, we stress the concept that, as described in Section 4.2, our scheme is resistant to such attacks and, therefore, no damage can be done by them to our reputation model. However, our approach natively implements a countermeasure to these attacks as the reputation of nodes carrying out them will be negatively impacted and, hence, they will be promptly isolated from the system. To test this additional capability of our solution, we focused on two common attacks in this category, namely Self-promoting and Bad-mouthing attacks. Specifically, starting again from the dataset above, we simulated the presence of different attackers controlling target nodes on the network and carrying out the attacks above. Moreover, we considered different attack intensity levels and studied the variation of the reliability score for these nodes. In particular, we organized this experiment into consecutive iterations during which attacked nodes are involved in the consensus mechanism and attack it with three different intensity levels: *Low*, *Medium*, and *High*. The attack intensity level depends on the variation between the real trust score of the target node (as correctly detected by the nodes in the paths of the consensus set) and the score forged by the attacker. In our experiment, we set a variation from 10% to 25% for a *low* intensity attack, from 25% to 50% for a *medium* intensity attack, and from 50% to 75% for a *high* intensity attack.

After each iteration, we computed the reliability score variation for the attacked nodes. We report the averaged results in Fig. 7, in which the dotted horizontal line indicates a standard limit threshold (half of the maximum reliability value) under which the node can be considered attacked and isolated from the system.

By analyzing this figure, we can see that the variation of the reliability score is very accentuated for attacks with high and medium intensity, thus allowing our solution to isolate the attacking nodes after just two iterations with a permissive threshold set to half of the maximum reliability score. As for attacks with low intensity, we can see that the reliability score decreases with a lower slope, and, hence, our system would need up to four iterations to identify the attacks and isolate the corresponding node.

As a final result, we also analyzed the overhead introduced by the use of the Blockchain as a shared platform to record and link the information about trust relationships among nodes. In Fig. 8, we report the number of transactions (write operations) and reading operations required on average to detect a compromised node.

By inspecting this figure, we can see that, in our experimental setup, when passing from a *low* to a *high* intensity attack, the number of reading operations ranges from 20 to 39, while the number of transaction created ranges from 4 to 8, on average. Observe that, as will be clearer in the next section, a possible target Blockchain for our solution is IOTA. On the IOTA 2.0 Devnet Nectar, the average transaction throughput is 1,000 transactions per second, while the confirmation time for a set of transactions is around 10 s [73]. This means, that all the transactions of a single iteration of our approach (i.e., the operations to compute a variation in the reliability of a node) can be written to the Blockchain in less than 1 second and their final confirmation will be visible in the Blockchain in about 10 seconds, on average.

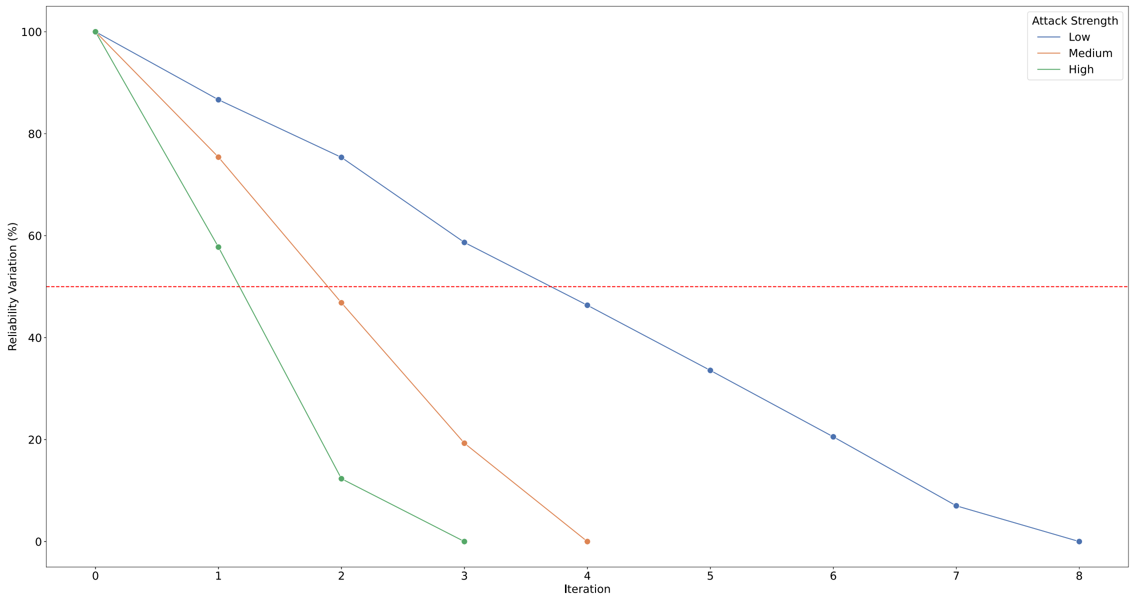

Fig. 7. Degradation of the reliability score for nodes carrying out attacks on the consensus mechanism.

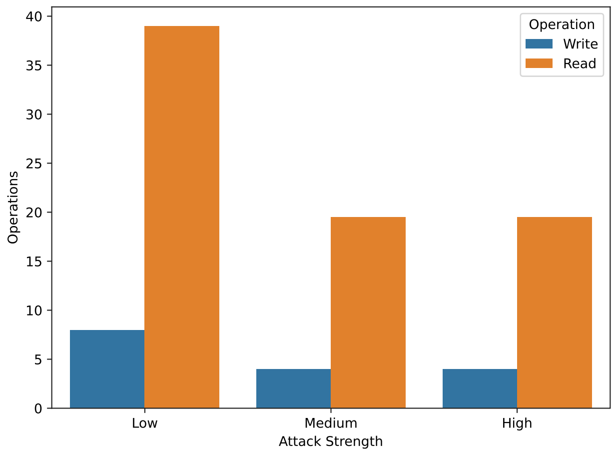

Fig. 8. Number of operations on the Blockchain required to identify a compromised node.

6. Integration of the blockchain in our solution

In this section, we focus on the role of the Blockchain and its integration into our proposal. As stated in Section 3.2, our solution exploits an underlying Blockchain layer as a shared ledger in which data about the trust and reputation of involved entities are stored. Specifically, the computation and the tracking of the evolution of reliability scores for each node involved in our approach are computed through a dedicated smart contract whose behavior is described in Section 3.4. Reliability values available in the Blockchain are, hence, used to control and protect the subsequent interactions among nodes. In particular, our solution enforces that nodes with a low (under a fixed threshold) reliability will not be involved in actions or engaged in future interactions by other peers. Also, as a special case, the Blockchain along with the reliability scores available in it is exploited to identify potential delegated nodes according to our secure delegation scheme described in Section 3.5. In this case, first, the reliability scores are used to identify valid candidates for delegation. Then, the Blockchain is used in conjunction with IPFS to exchange and control the necessary information to train or execute the inference of behavioral fingerprinting models.

However, despite its crucial role as a support tool in our solution, Blockchain technology along with its research challenges and open issues are orthogonal to our approach; this is actually a common strategy adopted by several research works in the literature [59]. Indeed, as initially stated in Section 3.2, we assume that any existing Blockchain can be used to implement our strategy. The only requirement is that it must explicitly support smart contract technology [74]. Moreover, in Section 4, we assume that the exploited Blockchain solution guarantees the standard security requirements already adopted for common Blockchain applications [58].

Therefore, also because our proposal does not aim at extending existing Blockchain solutions we do not consider vulnerabilities and possible direct attacks to it (see Security Property **SP.1**). Instead, in Section 4.2, we focus on attacks that can leverage the additional functionalities introduced by our approach and analyze the robustness of our security scheme against them. It is worth underlying that, although this may seem a limitation of our study, Blockchain security is currently under the spotlight of the research community and, therefore, a large number of security advancements are constantly studied and proposed [57,58]. In our work, we assume that the adopted Blockchain can benefit from and can be constantly updated with state-of-the-art solutions against known menaces.

Finally, we remark that in our approach we make explicit reference to a managed Blockchain dedicated to support our solution. In our basic design, we consider IoT nodes as light nodes (as opposed to the full nodes) of the Blockchain and, therefore, not involved in mining activities nor in the storage of the whole chain as typically done in the IoT context [75]. Generally, IoT devices cannot directly act as full nodes, and, hence, cope with the computational complexity and energy consumption of traditional Blockchain schemes. Indeed, the most popular Blockchain solutions are based on the famous Proof-of-Work paradigm, which is not suitable for IoT smart objects. Anyway, it is worth mentioning that, several approaches to building IoT-aware lightweight Blockchain have been proposed in the recent scientific literature [76–78]. One of the most discussed solutions in this setting is, for sure, the IOTA platform.[5] IOTA is based on a micro-transaction infrastructure and is specifically designed to support the IoT context. It is referred to as a more energy-efficient technology with respect to classical Blockchain schemes. The adoption of a lightweight Blockchain could also be a proper choice in our setting allowing for a more direct involvement of IoT nodes into the Blockchain management.

7. Conclusion

In the last years, we assisted in an enormous increase in the number and potentialities of IoT devices. From simple sensors/actuators to smarter nodes, all these actors enable the IoT network with complex monitoring, automation, and decision-making capabilities. Obviously, in this scenario, where IoT services and applications are intimately associated with people and are more and more autonomous, the issue of trust management becomes a major challenge. This paper makes a contribution in this setting, designing a complete framework to assess the trustworthiness of an object before contacting it. Our approach, based on collaboration and delegation, proceeds through two steps. At the initial stage of the network, behavioral models representing the conduct of every node are built thanks to a novel tiny machine learning algorithm suitable for limited devices. In the following fully operational state, every node is equipped with the possibility to detect possible variations or anomalies in the expected behavior of other objects and then decide to contact them. This feature is provided by a distributed consensus mechanism based on the concept of word-of-mouth between neighbors. Moreover, all the nodes, even the less smart ones can participate in our framework thanks to a secure delegation mechanism, according to which they can entrust the training of behavioral fingerprinting models to more powerful devices. Furthermore, Blockchain is used to store the reliability and trust scores related to the behavior of objects and to identify the best peers to contact to enable our collaborative approach.

The research directions taken in this paper can be considered as a starting point since we plan to make further investigations in this field in the future. For instance, behavioral fingerprinting of a group of objects can be analyzed to detect specific typologies of complex and distributed attacks in the network. A behavioral group fingerprinting solution can be seen as a machine learning model trained to predict the next possible distributed attack in the network.

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

We have used datasets already available online and properly referenced in our paper. The additional data built for our experiments can be made available on request.

Acknowledgments

This work was supported in part by the project SERICS (PE00000014) under the NRRP MUR program funded by the EU-NGEU, and by the PRIN 2022 Project "HOMEY: a Human-centric IoE-based Framework for Supporting the Transition Towards Industry 5.0" (code: 2022NX7WKE, CUP: F53D23004340006) funded by the European Union - Next Generation EU.

[5] www.iota.org

References

[1] Joseph Sifakis, System design in the era of IoT—meeting the autonomy challenge, 2018, arXiv preprint arXiv:1806.09846.
[2] Enrico Corradini, Serena Nicolazzo, Antonino Nocera, Domenico Ursino, Luca Virgili, A two-tier Blockchain framework to increase protection and autonomy of smart objects in the IoT, Comput. Commun. 181 (2022) 338–356.
[3] Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, Antonino Nocera, A privacy-preserving localization service for assisted living facilities, IEEE Trans. Serv. Comput. 13 (1) (2016) 16–29.
[4] Sureshkumar Selvaraj, Suresh Sundaravaradhan, Challenges and opportunities in IoT healthcare systems: A systematic review, SN Appl. Sci. 2 (1) (2020) 139.
[5] Serena Nicolazzo, Antonino Nocera, Domenico Ursino, Luca Virgili, A privacy-preserving approach to prevent feature disclosure in an IoT scenario, Future Gener. Comput. Syst. 105 (2020) 502–519.
[6] Anil Chacko, Thaier Hayajneh, Security and privacy issues with IoT in healthcare, EAI Endorsed Trans. Pervasive Health Technol. 4 (14) (2018) e2.
[7] HP, Hp study finds alarming vulnerabilities with Internet of Things (IoT) home security systems, 2015.
[8] Yigliana Alvarez, Miguel Angel Leguizamón-Páez, Tania J. Londoño, Risks and security solutions existing in the Internet of Things (IoT) in relation to big data, Ingeniería y competitividad.
[9] Alexander G. Eustis, The mirai botnet and the importance of IoT device security, in: 16th International Conference on Information Technology-New Generations, ITNG 2019, Springer, 2019, pp. 85–89.
[10] Sam Edwards, Ioannis Profetis, Hajime: Analysis of a decentralized internet worm for IoT devices, Rapidity Netw. 16 (2016) 1–18.
[11] Vikas Hassija, Vinay Chamola, Vikas Saxena, Divyansh Jain, Pranav Goyal, Biplab Sikdar, A survey on IoT security: Application areas, security threats, and solution architectures, IEEE Access 7 (2019) 82721–82743.
[12] Laura Rafferty, Farkhund Iqbal, Saiqa Aleem, Zhihui Lu, Shih-Chia Huang, Patrick C.K. Hung, Intelligent multi-agent collaboration model for smart home IoT security, in: 2018 IEEE International Congress on Internet of Things, ICIOT, IEEE, 2018, pp. 65–71.
[13] Jeffrey Voas, Richard Kuhn, Phillip Laplante, Sophia Applebaum, Internet of things (IoT) trust concerns, NIST Tech. Rep 1 (2018) 1–50.
[14] Hanan Aldowah, Shafiq Ul Rehman, Irfan Umar, Trust in IoT systems: A vision on the current issues, challenges, and recommended solutions, Adv. Smart Soft Comput. (2021) 329–339.
[15] Tigist Abera, N. Asokan, Lucas Davi, Farinaz Koushanfar, Andrew Paverd, Ahmad-Reza Sadeghi, Gene Tsudik, Things, trouble, trust: On building trust in IoT systems, in: Proceedings of the 53rd Annual Design Automation Conference, 2016, pp. 1–6.
[16] Salma Abdalla Hamad, Wei Emma Zhang, Quan Z. Sheng, Surya Nepal, IoT device identification via network-flow based fingerprinting and learning, in: 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE, IEEE, 2019, pp. 103–111.
[17] Kahraman Kostas, Mike Just, Michael A. Lones, Iotdevid: A behaviour-based fingerprinting method for device identification in the IoT, 2021, arXiv preprint arXiv:2102.08866.
[18] Marco Ferretti, Serena Nicolazzo, Antonino Nocera, H2O: Secure Interactions in IoT via Behavioral Fingerprinting, Future Internet 13 (5) (2021) 117.
[19] Alberico Aramini, Marco Arazzi, Tullio Facchinetti, Laurence S.Q.N. Ngankem, Antonino Nocera, An enhanced behavioral fingerprinting approach for the Internet of Things, in: 2022 IEEE 18th International Conference on Factory Communication Systems, WFCS, IEEE, 2022, pp. 1–8.
[20] Minhaj Ahmad Khan, Khaled Salah, IoT security: Review, blockchain solutions, and open challenges, Future Gener. Comput. Syst. 82 (2018) 395–411.
[21] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N. Asokan, Ahmad-Reza Sadeghi, DÏoT: A federated self-learning anomaly detection system for IoT, in: 2019 IEEE 39th International Conference on Distributed Computing Systems, ICDCS, IEEE, 2019, pp. 756–767.
[22] H. Tan, G. Tsudik, S. Jha, MTRA: Multiple-tier remote attestation in IoT networks, in: 2017 IEEE Conference on Communications and Network Security, CNS, IEEE, 2017, pp. 1–9.
[23] B. Kuang, A. Fu, S. Yu, G. Yang, M. Su, Y Zhang, ESDRA: An efficient and secure distributed remote attestation scheme for IoT swarms, IEEE Internet Things J. 6 (5) (2019) 8372–8383.
[24] C. Karlof, N. Sastry, D. Wagner, TinySec: A link layer security architecture for wireless sensor networks, in: Proc. of the International Conference on Embedded Networked Sensor Systems, SenSys'04, ACM, Baltimore, MD, USA, 2004, pp. 162–175.
[25] J. Chen, Z. Tian, X. Cui, L. Yin, X. Wang, Trust architecture and reputation evaluation for Internet of Things, J. Ambient Intell. Humaniz. Comput. 10 (8) (2019) 3099–3107.
[26] Taisuk Suh, Youngho Cho, An enhanced trust mechanism with consensus-based false information filtering algorithm against bad-mouthing attacks and false-praise attacks in wsns, Electronics 8 (11) (2019) 1359.
[27] Tayyab Khan, Karan Singh, Khaleel Ahmad, Khairol Amali Bin Ahmad, A secure and dependable trust assessment (SDTS) scheme for industrial communication networks, Sci. Rep. 13 (1) (2023) 1910.
[28] Bruhadeshwar Bezawada, Indrakshi Ray, Indrajit Ray, Behavioral fingerprinting of Internet-of-Things devices, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 11 (1) (2021) e1337.
[29] D. Ursino, L. Virgili, Humanizing IoT: defining the profile and the reliability of a thing in a multi-IoT scenario, in: Toward Social Internet of Things (SIoT): Enabling Technologies, Architectures and Applications, Springer, 2020, pp. 51–76.
[30] Sandhya Aneja, Nagender Aneja, Md Shohidul Islam, IoT device fingerprint using deep learning, in: 2018 IEEE International Conference on Internet of Things and Intelligence System, IOTAIS, IEEE, 2018, pp. 174–179.
[31] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, Sasu Tarkoma, IoT sentinel: Automated device-type identification for security enforcement in IoT, in: 2017 IEEE 37th International Conference on Distributed Computing Systems, ICDCS, IEEE, 2017, pp. 2177–2184.
[32] V. Thangavelu, D.M. Divakaran, R. Sairam, S.S. Bhunia, M. Gurusamy, DEFT: A distributed IoT fingerprinting technique, IEEE Internet Things J. 6 (1) (2018) 940–952.
[33] R. Di Pietro, X. Salleras, M. Signorini, E. Waisbard, A blockchain-based trust system for the Internet of Things, in: Proc. of the ACM International Symposium on Access Control Models and Technologies, SACMAT'18, ACM, Indianapolis, IN, USA, 2018, pp. 77–83.
[34] Mohamed Tahar Hammi, Badis Hammi, Patrick Bellot, Ahmed Serhrouchni, Bubbles of trust: A decentralized blockchain-based authentication system for IoT, Comput. Secur. 78 (2018) 126–142.
[35] Volkan Dedeoglu, Raja Jurdak, Guntur D. Putra, Ali Dorri, Salil S. Kanhere, A trust architecture for blockchain in IoT, in: Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, 2019, pp. 190–199.
[36] Michele Nitti, Roberto Girau, Luigi Atzori, Trustworthiness management in the social Internet of Things, IEEE Trans. Knowl. Data Eng. 26 (5) (2013) 1253–1266.
[37] Luigi Atzori, Antonio Iera, Giacomo Morabito, Michele Nitti, The social Internet of Things (SIoT)–when social networks meet the Internet of Things: Concept, architecture and network characterization, Comput. Netw. 56 (16) (2012) 3594–3608.
[38] S. Ganeriwal, R. Kumar, C.C. Han, S. Lee, M.B. Srivastava, Location & Identity Based Secure Event Report Generation for Sensor Networks, NESL Technical Report, 2004.
[39] Song Han, Huizi Mao, William J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2015, arXiv preprint arXiv:1510.00149.

[40] Gary White, Andrei Palade, Christian Cabrera, Siobhán Clarke, IoTPredict: Collaborative QoS prediction in IoT, in: 2018 IEEE International Conference on Pervasive Computing and Communications, PerCom, IEEE, 2018, pp. 1–10.
[41] Qingyang Zhang, Hong Zhong, Weisong Shi, Lu Liu, A trusted and collaborative framework for deep learning in IoT, Comput. Netw. 193 (2021) 108055.
[42] Alfonso Panarello, Nachiket Tapas, Giovanni Merlino, Francesco Longo, Antonio Puliafito, Blockchain and IoT integration: A systematic survey, Sensors 18 (8) (2018) 2575.
[43] Ali Dorri, Salil S. Kanhere, Raja Jurdak, Towards an optimized blockchain for IoT, in: 2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation, IoTDI, IEEE, 2017, pp. 173–178.
[44] Oscar Novo, Blockchain meets IoT: An architecture for scalable access management in IoT, IEEE Internet of Things J. 5 (2) (2018) 1184–1195.
[45] Serguei. Popov, Q. Lu, Iota: Feeless and free, 2019, IEEE Blockchain Technical Briefs.
[46] Lachit Dutta, Swapna Bharali, Tinyml meets IoT: A comprehensive survey, Internet Things 16 (2021) 100461.
[47] Jawad Ali, Ahmad Sharafidz Khalid, Shahrulniza Musa Eiad Yafi1, Waqas Ahmed, Towards a secure behavior modeling for IoT networks using blockchain, in: CEUR Workshop Proceedings, vol. 2486, 2019, pp. 244–258.
[48] Kyle Haefner, Indrakshi Ray, Complexiot: Behavior-based trust for IoT networks, in: First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, TPS-ISA, 2019.
[49] Rasim Alguliyev, Yadigar Imamverdiyev, Lyudmila Sukhostat, Cyber-physical systems and their security issues, Comput. Ind. 100 (2018) 212–223.
[50] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Tiezhen Wang, et al., Tensorflow lite micro: Embedded machine learning for tinyml systems, Proc. Mach. Learn. Syst. 3 (2021) 800–811.
[51] TensorFlow, TensorFlow Lite API Reference, 2022.
[52] Juan Benet, Ipfs-content addressed, versioned, p2p file system, 2014, arXiv preprint arXiv:1407.3561.
[53] F. Hendrikx, K. Bubendorfer, R. Chard, Reputation systems: A survey and taxonomy, J. Parallel Distrib. Comput. 75 (2015) 184–197.
[54] Ronald Cramer, Rosario Gennaro, Berry Schoenmakers, A secure and optimally efficient multi-authority election scheme, Eur. Trans. Telecommun. 8 (5) (1997) 481–490.
[55] Aneta Zwierko, Zbigniew Kotulski, A light-weight e-voting system with distributed trust, Electron. Notes Theor. Comput. Sci. 168 (2007) 109–126.
[56] Cristina Rottondi, Alessandro Panzeri, Constantin Tchetnkou Yagne, Giacomo Verticale, Detection and mitigation of the eclipse attack in chord overlays, Int. J. Comput. Sci. Eng. 13 (2) (2016) 111–121.
[57] Satpal Singh Kushwaha, Sandeep Joshi, Dilbag Singh, Manjit Kaur, Heung-No Lee, Systematic review of security vulnerabilities in ethereum blockchain smart contract, IEEE Access (2022).
[58] Saurabh Singh, A.S.M. Sanwar Hosen, Byungun Yoon, Blockchain security attacks, challenges, and solutions for the future distributed IoT network, IEEE Access 9 (2021) 13938–13959.
[59] Muhammad Nasir Mumtaz Bhutta, Amir A. Khwaja, Adnan Nadeem, Hafiz Farooq Ahmad, Muhammad Khurram Khan, Moataz A. Hanif, Houbing Song, Majed Alshamari, Yue Cao, A survey on blockchain technology: evolution, architecture and security, IEEE Access 9 (2021) 61048–61073.
[60] Sheikh Mohammad Idrees, Mariusz Nowostawski, Roshan Jameel, Ashish Kumar Mourya, Security aspects of blockchain technology intended for industrial applications, Electronics 10 (8) (2021) 951.
[61] Y. Yu, K. Li, W. Zhou, P. Li, Trust mechanisms in wireless sensor networks: Attack analysis and countermeasures, J. Netw. Comput. Appl. 35 (3) (2012) 867–880.
[62] S. Ganeriwal, L.K. Balzano, M.B. Srivastava, Reputation-based framework for high integrity sensor networks, ACM Trans. Sensor Netw. 4 (3) (2008) 1–37.
[63] N. Labraoui, M. Gueroui, L. Sekhri, A risk-aware reputation-based trust management in wireless sensor networks, Wirel. Pers. Commun. 87 (3) (2016) 1037–1055.
[64] Ahmed S. Abu Daia, Rabie A. Ramadan, Magda B. Fayek, AETiC AETiC, Sensor networks attacks classifications and mitigation, Ann. Emerg. Technol. Comput. (AETiC).
[65] Faisal Hussain, Syed Ghazanfar Abbas, Muhammad Husnain, Ubaid U. Fayyaz, Farrukh Shahzad, Ghalib A. Shah, IoT DoS and DDoS attack detection using resnet, in: 2020 IEEE 23rd International Multitopic Conference, INMIC, IEEE, 2020, pp. 1–6.
[66] Zubair A. Baig, Surasak Sanguanpong, Syed Naeem Firdous, Tri Gia Nguyen, Chakchai So-In, et al., Averaged dependence estimators for DoS attack detection in IoT networks, Future Gener. Comput. Syst. 102 (2020) 198–209.
[67] Nada Abughazaleh, R. Bin, Mai Btish, Dos attacks in IoT systems and proposed solutions, Int. J. Comput. Appl. 176 (33) (2020) 16–19.
[68] Mohammad M. Shurman, Rami M. Khrais, Abdulrahman A. Yateem, IoT denial-of-service attack detection and prevention using hybrid IDs, in: 2019 International Arab Conference on Information Technology, ACIT, IEEE, 2019, pp. 252–254.
[69] Ayyoob Hamza, Hassan Habibi Gharakheili, Theophilus A. Benson, Vijay Sivaraman, Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity, in: Proceedings of the 2019 ACM Symposium on SDN Research, 2019, pp. 36–48.
[70] Ville Satopaa, Jeannie Albrecht, David Irwin, Barath Raghavan, Finding a kneedle in a haystack: Detecting knee points in system behavior, in: 2011 31st International Conference on Distributed Computing Systems Workshops, IEEE, 2011, pp. 166–171.
[71] Somansh Kumar, Ashish Jasuja, Air quality monitoring system based on IoT using raspberry pi, in: 2017 International Conference on Computing, Communication and Automation, ICCCA, IEEE, 2017, pp. 1341–1346.
[72] R. Kumar, M. Pallikonda Rajasekaran, An IoT based patient monitoring system using raspberry pi, in: 2016 International Conference on Computing Technologies and Intelligent Data Engineering, ICCTIDE'16, IEEE, 2016, pp. 1–4.
[73] Nathan Sealey, Adnan Aijaz, Ben Holden, Iota tangle 2.0: Toward a scalable, decentralized, smart, and autonomous IoT ecosystem, in: 2022 International Conference on Smart Applications, Communications and Networking, SmartNets, IEEE, 2022, pp. 01–08.
[74] Shafaq Naheed Khan, Faiza Loukil, Chirine Ghedira-Guegan, Elhadj Benkhelifa, Anoud Bani-Hani, Blockchain smart contracts: Applications, challenges, and future trends, Peer-to-peer Netw. Appl. 14 (2021) 2901–2925.
[75] Santeri Paavolainen, Christopher Carr, Security properties of light clients on the ethereum blockchain, IEEE Access 8 (2020) 124339–124358.
[76] M. Rehman, N. Javaid, M. Awais, M. Imran, N. Naseer, Cloud based secure service providing for IoTs using blockchain, in: Proc. of the IEEE Global Communications Conference, GLOBCOM 2019, Puako, Hawaii, USA, 2019, pp. 1–7.
[77] A.D. Dwivedi, G. Srivastava, S. Dhar, R. Singh, A decentralized privacy-preserving healthcare blockchain for IoT, Sensors 19 (2) (2019) 326.
[78] W.F. Silvano, R. Marcelino, Iota Tangle: A cryptocurrency to communicate Internet-of-Things data, Future Gener. Comput. Syst. 112 (2020) 307–319.