From Trustworthy Data to Trustworthy IoT: A Data Collection Methodology Based on Blockchain

CLAUDIO A. ARDAGNA, Università degli Studi di Milano
RASOOL ASAL, EBTIC, Khalifa University of Science, Technology and Research
ERNESTO DAMIANI*, EBTIC, Khalifa University of Science, Technology and Research
NABIL EL IOINI, Free University of Bozen
MEHDI ELAHI, University of Bergen
CLAUS PAHL, Free University of Bozen

Internet of Things (IoT) is composed of physical devices, communication networks, and services provided by edge systems and over-the-top applications. IoT connects billions of devices that collect data from the physical environment, which are pre-processed at the edge and then forwarded to processing services at the core of the infrastructure, on top of which cloud-based applications are built and provided to mobile end users. IoT comes with important advantages in terms of applications and added value for its users, making their world smarter and simpler. These advantages, however, are mitigated by the difficulty of guaranteeing IoT trustworthiness, which is still in its infancy. IoT trustworthiness is a must especially in critical domains (e.g., health, transportation) where humans become new components of an IoT system and their life is put at risk by system malfunctioning or breaches. In this paper, we put forward the idea that trust in IoT can be boosted if and only if its automation and adaptation processes are based on trustworthy data. We therefore depart from a scenario that considers the quality of a single decision as the main goal of an IoT system, and consider the trustworthiness of collected data as a fundamental requirement at the basis of a trustworthy IoT environment. We therefore define a methodology for data collection that filters untrusted data out according to trust rules evaluating the status of the devices collecting data and the collected data themselves. Our approach is based on blockchain and smart contracts, and collects data whose trustworthiness and integrity are proven over time. The methodology balances trustworthiness and privacy, and is experimentally evaluated in real-world and simulated scenarios using Hyperledger fabric blockchain.

CCS Concepts: • **Information systems** → **Computing platforms**; **Trust**; *Information integration*; • **Computer systems organization** → **Distributed architectures**; • **Security and privacy** → *Security services.*

Additional Key Words and Phrases: Blockchain, Internet of Things, Trustworthiness

ACM Reference Format:
Claudio A. Ardagna, Rasool Asal, Ernesto Damiani, Nabil El Ioini, Mehdi Elahi, and Claus Pahl. 2019. From Trustworthy Data to Trustworthy IoT: A Data Collection Methodology Based on Blockchain. *ACM Transactions on Cyber-Physical Systems* 1, 1, Article 1 (January 2019), 26 pages.

*Ernesto Damiani is also with Università degli Studi di Milano

Authors' addresses: Claudio A. Ardagna, claudio.ardagna@unimi.it, Università degli Studi di Milano, Milano, Italy, Rasool Asal, rasool.asal@bt.com, EBTIC, Khalifa University of Science, Technology and Research, Abu Dhabi, UAE, ; Ernesto Damiani, ernesto.damiani@ku.ac.ae, EBTIC, Khalifa University of Science, Technology and Research, Abu Dhabi, UAE, ; Nabil El Ioini, nelioini@unibz.it, Free University of Bozen, Bolzano, Italy, Mehdi Elahi, Mehdi.Elahi@uib.no, University of Bergen, Bergen, Norway, Claus Pahl, Claus.Pahl@unibz.it, Free University of Bozen, Bolzano, Italy,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.
XXXX-XXXX/2019/1-ART1 $15.00
https://doi.org/

1 INTRODUCTION

Internet of Things (IoT) can be defined as "*the networked interconnection of everyday objects, equipped with ubiquitous intelligence*" [32]. From the original launch of the IoT idea, more than 10 years ago, technology has rapidly progressed and several aspects of IoT came to maturity and entered commercial offerings. We are increasingly moving from a centralized approach where computations are done at the core of the network (e.g., cloud), to a scenario where analytics and knowledge extraction are partially done at the edge near the physical environment and sensors where data are collected. This scenario will be put to the extreme by the exponential growth of connected devices (from minuscule sensors to bigger machines), which, according to Intel,[1] are expected to reach 200 billions by 2020.

The existence of billions of resource-constrained devices connected to the Internet introduces fundamental risks that can threaten users' life and personal sphere. Devices are so pervasive that humans have just become another component of the system, with all risks and unpredictability introduced when human decisions are put in the automation loop. A wealth of services in different domains, such as smart vehicles, smart buildings, e-health, are distributed on the basis of data collected by devices. In this context, system automation and adaptation is based on the assumptions that the system and its devices behave correctly, and collected data are trustworthy [3]. Amit Sheit in [27] maps the Data, Information, Knowledge and Wisdom (DIKW) hierarchy [22] to IoT. The hierarchy is used to "*contextualize data, information, knowledge, and sometimes wisdom, with respect to one another and to identify and describe the processes involved in the transformation of an entity at a lower level in the hierarchy*" [22]. The implicit assumption is that data have a sufficient level of trustworthiness to create information, and in turn knowledge and wisdom. Recent work followed this assumption in the definition of new assurance techniques [3, 6], as well as new automation solutions and adaptive techniques [30], where collected *data* are first aggregated (*information*) and then transformed (*knowledge*) to take a decision (*wisdom*). This assumption is however not sound when a plethora of devices are used to collect data, and might bring to scenarios where wrong data results in wrong decisions and, in turn, untrusted services/applications. Resource-constrained devices are in fact more subject to hacking or malfunctioning, and becomes a main driver towards system manipulation.

This paper shifts the focus from the need of trustworthy automation and adaptation to the need of trustworthy data at the basis of provable automation and adaptation processes. The idea is to provide an approach that complements existing trustworthy decision processes with a methodology for collecting trustworthy data, identifying untrustworthy data sources due to sensor malfunctioning or configuration errors also including those configurations that can indirectly affect the correctness of sensor communications (e.g., weak encryption algorithms). The intuition behind our approach is that the more trustworthy data, the higher the decision accuracy. Our approach provides a novel methodology for trustworthy data collection at the basis of trustworthy IoT, where data collected from devices are first checked for correctness and then stored in the blockchain using specific smart contracts. Blockchain can play an important role in IoT trustworthiness evaluation since *i)* it provides a robust decentralized architecture that could stand against failures and attacks, *ii)* all participants in the network share the same truth (data) and contributes to support the honest nodes, weakening the possibility of data manipulation, *iii)* it relies on Public Key encryption, which is expensive to crack and guarantees to identify all participants. Trustworthiness of data is evaluated at two levels: *i) syntactic level* where physical parameters and configurations of devices (e.g., firmware version, direction of a camera, data format and syntax) are used to filter out data produced by untrusted devices, *ii) semantic level* where system/device behaviors are modeled (from

[1] https://www.intel.com/content/www/us/en/internet-of-things/infographics/guide-to-iot.html

statistical parameters to complex machine learning models) to filter out untrustworthy/incorrect data (e.g., a fire alarm sensor sending an alarm when temperature is less than 20 degrees Celsius).

This paper develops on our previous work [2] by providing *i)* an architecture for trustworthy data collection based on blockchain and smart contracts (Section 4), *ii)* a methodology for the assessment of the trustworthiness of collected data based on syntactic and semantics rules (Sections 5 and 6), *iii)* different processes for collecting, verifying, and storing trustworthy data, which balance trustworthiness and privacy (Section 7). We further evaluate the performance of our approach (Section 8) and its quality in a simulated scenario considering data collected from smart homes and in a real-world scenario considering roaming data collected from the switching infrastructure of ETISALAT, the largest telecommunication corporation in the Gulf Cooperation Council (Section 9).

2 PROBLEM STATEMENT AND RELATED WORK

The advantages introduced by smart IoT systems clash with the new risks introduced in (critical) domains (e.g., health, transportation) where human safety depends on IoT and is threatened by it. These risks are even worse when autonomic decisions and adaptations (e.g., [6, 14]) are employed for increasing the system performance and quality. We put forward the idea that the success of IoT and edge systems cannot depart from an open, protocol-neutral baseline solution for IoT trustworthiness. This solution must be grounded on a data collection methodology that verifies data trustworthiness. Trustworthy data are in fact the cornerstone for implementing safe autonomic and adaptive processes at the basis of IoT system functioning. Wrong decisions, such as a smoke detector not detecting smoke properly and impairing the correct functioning of a fire alarm, can result in an incalculable damage to users.

Current literature (e.g., [35]) seems to ignore this problem and usually assumes trustworthy data, or at least that "*superiority in numbers is the most important factor in the result of a combat (cit. Clausewitz)*", meaning that the availability of a huge amount of devices should support trustworthy decisions also in case a not-negligible part misbehaves. Current autonomic and adaptive systems then take a decision on data directly coming from sensors with no filtering. This assumption introduces important limitations when targeting complex IoT systems, where autonomic and adaptation decisions are taken at run time, as follows.

- *Hybrid and complex systems.* Current hybrid systems connect cloud systems at the center and smart devices at the periphery, via edge networks. The trustworthiness of devices and, in turn, of their data cannot be simply based on the assumption that *superiority in numbers is the most important factor*. The trustworthiness of a sensor measurement can be evaluated by observing the values of the other sensors. A proper assessment must satisfy heterogeneous requirements coming from the different parts of the system.
- *Untrusted (micro) providers.* Hybrid systems rely on data continuously collected by a multitude of devices, which are intrinsically unreliable and under the control of many untrusted providers. A proper assessment must depart from any form of trust on remote providers.
- *Untrustworthy data collection.* Traditional autonomic and adaptive processes are often driven by untrusted/unverified data that are accepted on the basis of the provider reputation. In addition, collection mechanisms are often blurred into the target system, imposing a take or leave approach. These assumptions can hinder the soundness of the system itself, since smart devices are owned by micro providers, they can be hacked or simply fail. A proper assessment should be grounded on a standard and trustworthy data collection.
- *Untrustworthy decision.* Traditional autonomic and adaptive processes consider the target system as a whole, introducing decision that might interfere with the behavior of a specific subsystem. Hybrid systems introduce the need of taking a bottom-up approach, where

trustworthy data on local devices/services/processes are first collected and evaluated to generate local claims, and local claims are then composed to provide global assessment on the whole system [4].

- *Unverifiable decision.* Traditional autonomic and adaptive systems aim to maximize the quality of a decision (e.g., scalability). It is often difficult to prove/audit the correctness of such decisions. This problem points to the need of an accountable trustworthy data collection.

The need of assessing data trustworthiness clearly emerges in different critical scenarios.

First, the advent of cloud and IoT make the need of evaluating non-functional properties (e.g., performance, security) of current systems more stringent than ever. Assurance techniques can contribute to address this issue, by increasing the confidence that a system behaves as expected. Their definition recently targeted the cloud (e.g., [3, 6, 9]). Concerning IoT, the research community has mainly focused on new security protocols and techniques to support the CIA (Confidentiality, Integrity, Availability) triad (e.g., [8, 33]), while research on assurance is at an early stage. Preliminary work focused on defining new assurance architectures for IoT. Ardagna et al. [4] first discussed challenges in the design and development of assurance techniques for IoT, proposing a conceptual framework and architecture for IoT security assurance evaluation. Sato et al. [24] investigated the problem of trust establishment in IoT and proposed an architecture for evaluating "*area-wise trust*", where the trust level considers device identification, monitoring of device behaviors, device connection processes and protocols. Taherizadeh et al. [31] presented a survey on the monitoring of self-adaptive applications based on decentralized edge computing. The proposed approach assumes that collected data are trustworthy and do not affect the precision of the assessment, denying possible *false-positive* scenarios resulting in a wrong assessment.

Other approaches based on remote attestation focused on ascertaining the legitimate operation of potential untrusted devices and on establishing trust in IoT devices, by detecting malware installed on them. Ambrosin et al. [1] presented an architecture for a scalable and secure IoT management. The approach relies on two protocols that send control messages and monitor the status of IoT devices. Al-Hamadi and Rani et al. [20] presented a game theory-based approach to improve IoT sensor trustworthiness in the automotive domain, by creating clusters of sensors to identify illegitimate and malicious nodes. Although the above work focuses on the assessment of a system trustworthiness, they do not consider the problem of executing a management/autonomic function on the basis of untrustworthy data. This could result in scenarios where a malicious user injects fake data to manipulate the correct execution of a given system workflow. In addition, the need of evaluating the trustworthiness of collected data clearly emerged in the context of forensics science and has been initially dealt with by defining a systematic and reliable methodology for data collection and analysis [10]. Some solutions based on blockchain have been also proposed to guarantee availability, integrity, and verifiability of collected data (e.g., [5, 16, 19]). Their goal was to show the feasibility of using blockchain to guarantee integrity and traceability of digital forensics evidence, while not focusing on the quality and accuracy of stored results.

Some effort has been devoted to IoT trust management and evaluation. Jayasinghe et al. [11] presented a data-centric evaluation and precision framework that defines a set of trust metrics. Machine learning is used to compute the trust prediction model based on the data collected from the involved entities. Maxim et al. [21] proposed a data-oriented trust model for Ephemeral ad-hoc Networks. The model establishes an initial trust relationship between the involved parties and defines a set of metrics such as time freshness and location relevance, which are used by Bayesian inference and Dempster-Shafer Theory to evaluate the collected evidence. Yang et al. [34] proposed a blockchain-based trust management system in Vehicular Networks. The main idea relies on validating messages received from neighboring vehicles, by classifying the messages based on

their type and the vehicles location. Dai et al. [7] introduced a trustworthiness framework that considers both data and data providers. The main idea is to assign different trust scores to data and data provider based on four factors: *i) data similarity*, *ii) path similarity*, *iii) data conflicts*, *iv) data deduction*. The scores are calculated using a similarity function to put the similar entities and data points at the same level of trust. Similarly to [7], Suhail et al. [30] proposed a lightweight provenance schema that does not add high overhead in an IoT environment. The limitation of this approach is that it does not consider the messages content (e.g., wrong measurements). As sensors data is critical to the decision making process, the authors in [13] developed a trust model for event selection in wireless sensor networks. The approach implements a data fault detection and data reconstruction based on spatio-temporal data correlation and attribute data correlation. In general, although the work done in the context of trust management and evaluation in IoT represents a good starting point, it fails to provide a general-purpose methodology for trustworthy data collection, at the basis of a trustworthy decision process in IoT. Most of the work either focus on specific scenarios (e.g., vehicular ad-hoc networks) or on specific requirements (e.g., data freshness, location relevance), often building on assumptions such as superiority in numbers and trustworthiness of the data provider.

To conclude, a considerable research attention has been devoted to the use of blockchain technology to improve the security and scalability of IoT devices. Lin et al. [15] implemented and tested a fully decentralized solution for IoT data integrity. Their goal was to ensure integrity of data from IoT devices with a blockchain-based Data Integrity as a Service (DIaaS) framework, where Integrity Management Service (IMS) is available as part of cloud services. Özyılmaz and Yurdakul [18] used blockchain to build a decentralized IoT platform standardizing the way data transfers are handled among IoT devices. At the same time, Blockchain has been heavily used in combination with IoT to manage devices identity and data provenance, such as in [17], where blockchain is used to identify/authenticate IoT devices. Shi et al. [28] proposed triple-trusting architecture (SLTA), a framework for *i)* checking the collected data before injecting them in the digital world, *ii)* ensuring provenance of data, and *iii)* guaranteeing IoT identity and data authenticity. Similarly, Sigwart et al. [29] relied on blockchain to track IoT data provenance. The study defined different provenance granularities, while not considering inconsistencies in the data itself.

To address the above limitations, we define a methodology for the collection and verification of trustworthy data, which is the basis for any trustworthy IoT processes and systems. Our approach is based on blockchain and smart contracts, and collects data whose trustworthiness and integrity are proven over time. Differently from existing solutions, our approach links the data to the way in which they are collected and verified, as well as to the status and configurations of the collecting device. Untrustworthy data are then filtered out and not given as input to the autonomic and adaptive system. IoT Trust, the core of our approach, is presented in Sections 5, 6, and 7.

3 REFERENCE SCENARIO

Our reference scenario considers a *smart home*, where ICT technologies are employed to automate many tasks and improve users' quality of life. Smart homes are composed of a heterogeneous infrastructure of sensors and actuators where enhanced lighting, energy, heating, air conditioning, and physical security systems are integrated together to increase users' experience (e.g., security, comfort, convenience). Sensors are deployed at different system layers to connect people with technology, and collect precise and accurate data on the environment status, such as people locations, energy usage profile, and emergency situations; actuators and controllers use these data to manage and adapt the homes behavior according to predefined rules and policies. The embedded intelligence in smart homes can *i)* provide users with predictive maintenance, for instance, to avoid spending a cold day without heating or manage refill of restroom supplies, *ii)* interact with smart grids for

better energy consumption and management of exceptional events (e.g., increasing availability of critical services in case of power failures), *iii)* support services for emergency management. In this scenario, decisions rely on data collected from sensors. In some cases, a single sensor can be decisive (e.g., turn lights on when a person is detected); in other cases, decisions can only be made based on multi-sensor input (e.g., raise a fire alarm to the fire department). Data trustworthiness as a way to increase users' confidence that the services and devices behave as expected is a fundamental requirement and represents the cornerstone of any automation systems. Our reference scenario requires accurate and trustworthy data collected by sensors that hold properties of integrity, traceability, authentication, verifiability, security, and privacy, to name but a few. For example, smart homes rely on hidden sensitive data on the corresponding household/guest. These data carry sensitive information that can be used to infer people habits and behavior, and pose strict requirements on their management (*property privacy*). Trustworthy data on local devices can be used to produce local claims on the status of a given object/subsystem under evaluation (e.g., smart home power consumption). Such claims are at the basis of a process-wide assurance verification, where local claims are composed in process-wide global claims to verify the status of an IoT-enabled processes, spanning hybrid networks. In this case, the accuracy and provenance of data is paramount (properties integrity, traceability, and verifiability).

4 TRUSTWORTHY DATA COLLECTION ARCHITECTURE

Our trustworthy data collection architecture is composed of three main components that constantly monitor and verify data coming from smart devices (e.g., performance, usage). The data target of our collection process are pieces of information at the basis of any decision processes, which can be used to trigger an action (e.g., fire alarm) or prove a state (e.g., temperature value). Data can come from different sources (e.g., sensors, edge nodes, users), have different granularities (e.g., single sensor reading, aggregated data, decision), and be of different sensitivity levels (e.g., private or public data). For instance, every smart home is equipped with a set of power consumption sensors that measure how much power their appliances consume on an interval base or upon changes in the power consumption level. Once a day, sensor readings are aggregated to produce an evidence representing the average power consumed by the smart building. Data, possibly aggregated, supports system automation and automated decisions (e.g., buildings that have energy surplus would support possible power trading). Collected data are then verified and stored in the blockchain balancing trustworthiness, performance, and privacy requirements over time. To this aim, a set of dedicated smart contracts are used to program various checks and data verification before persisting the coming data. Checks include ensuring that the data sources are legitimate, trustworthy and have the correct access rights, as well as the data formatting and content are correct. Accordingly, our architecture is mapped on three layers, as follows.

- *Data Collection Layer.* It connects to device streams and collects all data relevant for taking a decision. Such data can be either measurements returned from the devices (e.g., a power consumption sensor) or metadata describing physical configurations of the devices. Each device digitally signs its readings before sending them to the data validation process.
- *Data Validation Layer.* It verifies the trustworthiness of collected data using the blockchain. Once data are sent back by smart devices, the smart contract managing the interaction with the blockchain first verifies the identity of the device registered in the blockchain. The data validity is then checked based on the type of smart device and external knowledge (see Section 5 for more details). For instance, data validity can depend on the data timestamps or on the status of a sensor (e.g., the installation of the latest firmware update), as well as on the modeling of the correct device behavior.

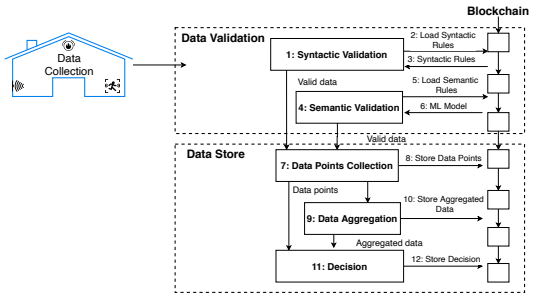


Fig. 1. Architecture Overview

- *Data Store Layer.* It stores trustworthy data on chain using a smart contract. Once the validation process is successfully executed, a transaction containing the new data is persisted on the blockchain. Different amounts of data can be stored in the blockchain, that is, single data points, aggregated data, or decisions taken on trustworthy data, depending on trustworthiness, performance, and privacy requirements (see Section 6 for more details).

Figure 1 illustrates the execution flow of our architecture. Data collection triggers the execution flow at predefined intervals or when certain events take place. Upon collecting data (phase 1), data validation process first validates every data point against a set of syntactic rules, modeling requirements on physical parameters and configurations specific to each device type (phases 1, 2, 3 in Figure 1). Data validation process then validates data points against semantic rules (phases 4, 5, 6 in Figure 1), employing machine learning techniques to find correlations between data points coming from different sources (e.g., readings from two neighboring noise sensors).[2] Our idea is that the problem of identifying untrustworthy data sources due to sensor malfunctioning or configuration errors can be attacked as a standard classification problem, rather than using complex deep learning approaches adopted to counteract poisoning attacks (e.g., [12, 26]). We note that syntactic and semantic rules are pre-loaded in the blockchain to increase the performance of the validation process. Data points returned by the data validation process are syntactically and/or semantically correct, and are then used to drive a trustworthy decision process. Data points can be first stored as is in the blockchain (phases 7 and 8 in Figure 1). Otherwise, if needed according to the specific use case, data points can be aggregated (phases 9 and 10 in Figure 1), and then used to take a decision or adapt the system (phase 11 and 12 in Figure 1). The activities and results of all phases, including syntactic validation, semantic validation, data aggregation, decision, can be executed/stored in the blockchain to increase traceability, transparency, and trustworthiness, at the price of a decreased privacy and performance. We will discuss all details our trustworthy data collection process in Section 5 (discussing smart contracts for data collection and validation), Section 6 (discussing smart contracts for data store), Section 7 (defining the trustworthy data collection process), and Section 7.4 (discussing the properties of our process).

5 DATA VALIDATION: IOT TRUST

Traditional holistic approaches to trust verification are not viable when heterogeneous and resource-constrained devices, fuzzy perimeters, and a multitude of micro-providers are involved (Section 2). We need to depart from approaches where data trustworthiness is linked to reputation, as well as from approaches where testing and monitoring techniques are used to evaluate each system

[2]We note that syntactic validation and semantic validation are optional and their adoption depends on the specific use case. For instance, when all validation rules are known a priori, the semantic validation can be omitted.

component. Rather, an atomistic approach based on trustworthy data collection can be the basis for implementing a trustworthy IoT environment, where trustworthy processes and decisions are employed. Data collected from each smart device must be first validated and then put into service only if a minimum amount of trust requirements are addressed.

In this section, we present *IoT Trust*, our methodology for the validation of data trustworthiness. We handle data validation in the blockchain using dedicated smart contracts. A smart contract is composed of a set of data structures modeling collected data, a set of functions that act on the data and verify trust requirements, and a set of emitted events. We define *IoT Trust* as a smart contract specified as follows.

Definition 5.1 (IoT Trust)**.** IoT Trust is a smart contract SC_v defined as a 3-tuple $(\mathcal{D}, \mathcal{R}, \mathcal{E})$ where:

- Data structure $\mathcal{D}$ defines a set $\{D_p\}$ of collected data points (Section 5.1);[3]
- Validation rules $\mathcal{R}$ models trust requirements to be addressed by trustworthy data (Section 5.2);
- Events $\mathcal{E}$ defines a set $\{E_1, \ldots, E_m\}$ of events emitted during contract execution (Section 5.3).

When a data point (or a set thereof) is collected, the smart contract is executed and associates a label trustworthy/untrustworthy with it. In the following, after presenting our data structure, we discuss two types of validation rules used to define trust requirements and corresponding events.

5.1 Data structure

The data structure captures the attributes needed for the validation of data trustworthiness. The basic piece of data is a measurement Δ, which describes a fact or a measurement coming from a smart device (from minuscule sensors to bigger machines). Each measurement is extended with metadata, that is, a set of attributes that describe the collected data. Examples of measurement metadata are timestamp and data owner. A measurement Δ is then formally defined as follows.

Definition 5.2 (Measurement Δ)**.** A measurement Δ is a 2-tuple $\Delta = (\delta, M_\delta)$, where δ is a device reading (a, v) pair, with a the measured attribute and v its value, and M_δ is a finite set of metadata pairs (m, v), with m the metadata of a and v its value.

In forensic science, to properly document the collected measurements, its metadata are enriched with a proof on the integrity of the measurement tools. To this end, we define data point dp as an augmentation of the measurement Δ with the device metadata, which represents information about the devices and their environment. Examples of device metadata are the version of the device firmware and the digital signature. A data point dp is then formally defined as follows.

Definition 5.3 (Data point dp)**.** Data point $dp \in \mathcal{D}$ is a 3-tuple $dp(\Delta, s, M_s)$, where Δ is a measurement, s is the device submitting the data point, and M_s is a finite set of device metadata pairs (m, v), with m the metadata of s and v its value.

The collected data can be kept at the data point granularity or contain a set of data points to provide a temporal view of data evolution.

Example 5.1. A smart home can be equipped with a camera recording any accesses. In this example, the data point contains the video stream (device reading), the timestamp and the data owner (measurement metadata), and physical parameters of the device such as camera orientation, firmware version, and frame rate (device metadata).

[3]When clear from the context, we will use D_p to denote a set of data points

5.2 Validation Rules

Validation rules $\mathcal{R}$ model trust requirements to be addressed by collected data points before being stored as trustworthy data on chain or used to take a decision. For instance, when a device reading is submitted, IoT trust compares its value against $\mathcal{R}$ to filter out untrustworthy data violating these rules. In this section, we define two validation rules, syntactic rules (Section 5.2.1) and semantic rules (Section 5.2.2), evaluating data points in Section 5.1. Data points include measurements (i.e., numerical values), measurement and device metadata (i.e., descriptive information), continuous data (i.e., stream of data points collected in an interval). We note that syntactic rules validate device metadata, while semantic rules validate the measurement (including measurement metadata). When clear from the context, we use metadata to refer to both device and measurement metadata.

5.2.1 Syntactic Rules. Syntactic rules introduce requirements on physical parameters and configurations of devices. They assume that trustworthy data can only be collected from trustworthy devices. For instance, semantic rules verify the format of data points and the firmware version of the device collecting them. They are implemented as threshold-based rules as follows.

Definition 5.4 ($\mathcal{R}_I$)**.** Function $\mathcal{R}_I$ is a membership function that takes as input a data point dp and returns as output a Boolean value indicating whether a data point is syntactically trustworthy (*true*) or not (*false*), according to a set of thresholds-based conditions c_i in the form (dp *op value*), with $op \in \{=, \neq, <, \leq, >, \geq\}$. It assumes the following form:

$$\mathcal{R}_I(dp) = \begin{cases} True & if \quad (c_1 \wedge c_2 \wedge \ldots \wedge c_n) \\ False & else \end{cases}$$

We note that these rules are used to filter out all data points that are syntactically wrong, such as for instance those data points received from a misconfigured sensor (e.g., using a vulnerable firmware). They are generally applied to device metadata M_s.

Example 5.2. Following our reference scenario, syntactic rules verify the correctness of devices installed in the kitchen of a smart home. For instance, syntactic rules can verify the firmware of household appliances, the range of the fridge temperature, the correct operation of a light bulb.

5.2.2 Semantic Rules. Semantic rules model the expected behavior of a single device or a collection thereof. They are generated from the historical data collected by target devices and used to evaluate the behavior of new data points. They employ machine learning techniques to generate a model of the expected behavior. This model represents the signature of trustworthy data points and is then used to classify the new incoming data points (in isolation or batch) as trustworthy or not according to their behavior. Formally, we consider a standard machine learning algorithm where a training phase is first used to generate the behavioral model and an inference phase uses the generated model to mark data points as trustworthy or not. The training phase takes as inputs a labeled data set T (training set) as a set of pairs of the form:

$$T = \{(dp_1, l_1), , \ldots, (dp_n, l_n)\}$$

where dp_i is a data point and l_i is the corresponding label $\{trustworthy, untrustworthy\}$. The output of the classification algorithm is a model $\mathcal{R}_D$ that given a data point dp (or a set thereof) as input returns a label l as output, as follows:

$$\mathcal{R}_D : DP \rightarrow L \quad such\ that \quad \mathcal{R}_D(dp) = l$$

We note that our discussion does not force the use of any machine learning algorithms; rather its selection is left to the domain administrator and aims to maximize the benefit in terms of performance and accuracy. We also note that simpler semantic rules using threshold-based conditions

(Definition 5.2.1) can be used. For instance, a rule can state that the room temperature cannot exceed 40 degrees in winter.

Example 5.3. Following our reference scenario, semantic rules try to uncover abnormal activities that diverge from the expected behavior. For instance, a cooking process taking place in the late evening but with light sensor off; a sport activity taking place in the late night with alarm sensors activated.

5.3 Events

Events are mechanisms put in place by IoT trust to emit changes in the blockchain state, according to those data passing syntactic and semantic checks, which can be used by external services/applications to take specific actions. Events are defined inside the smart contracts and are fired when certain conditions occur. Events can be emitted from all contracts depending on the specific scenario.

Example 5.4. The heating system can register to all events related to temperature changes; this way, if the temperature exceeds the pre-defined threshold (e.g., 40 degrees), an event of the form $\{TD, heating, off\}$ is emitted.

6 DATA STORE

We use blockchain as the data repository that contains all transactions for trustworthy data collection and validation. The state of a blockchain is represented by a $k{-}v$ data store $BS{:}k{\rightarrow}v$, where k is a key and v is an arbitrary sequence of data. In our context, v is the data collected at different levels of abstraction. The blockchain handles data store at various degrees using dedicated smart contracts, as formally defined below.

Definition 6.1 (Data Store)**.** Data Store is a smart contract SC_s defined as a 3-tuple $(\mathcal{D}, \mathcal{R}, \mathcal{E})$ where:

- Data structure $\mathcal{D}$ defines a set $\{D_p\}$ of trustworthy data points returned by IoT Trust in Section 5;
- Function calls $\mathcal{F}$ manipulates data before storing them on chain;
- Events $\mathcal{E}$ defines a set $\{E_1, \ldots, E_m\}$ of events emitted during contract execution.

Data structure $\mathcal{D}$ and events $\mathcal{E}$ are defined in Section 5, and consists of a set of data points in Definition 5.3 and events in Section 5.3, respectively. Data store provides three different store functions that are implemented at different granularities, supporting the store of simple data points, data aggregations, and decisions.

6.1 Data Point Store

It implements a function that stores trustworthy data points in the blockchain. It is defined as follows.

Definition 6.2 (*EC*)**.** Function Data Point Store *EC* takes as input a data point *dp* and stores it in the blockchain *BS*.

This function tracks all data points that satisfied the trust requirements (validation rules) on chain.

6.2 Data Aggregation Store

It implements a function that supports the calculation of specific metrics and stores them as aggregated data directly on chain. It is formally defined as follows.

Definition 6.3 (DA)**.** Function Data Aggregation Store DA: $D_p \times op \rightarrow \mathcal{A}$ takes as input a set D_p of data points $\{dp_1, dp_2, \ldots, dp_n\}$ within a specific time interval, an aggregation operator $op \in \{sum, average, min, max, count\}$, and produces as output the aggregated data $\mathcal{A}=op(D_p)$ that is then stored in blockchain BS.

This function calculates and tracks aggregated data on chain.

6.3 Decision Store

It implements a function that supports the calculation of a specific decision based on collected data points and data aggregation, and stores it in the blockchain. It is formally defined as follows.

Definition 6.4 (TD)**.** Function Decision Store TD: $\mathcal{D} \times df \rightarrow \mathcal{P}$ takes as input a set $\mathcal{D}$ of data points or aggregated data, a decision function df, and produces as output the decision $\mathcal{P}=df(\mathcal{D})$ that is stored in blockchain BS.

A decision function is then defined as follows.

Definition 6.5 (df)**.** Function df is a membership function that takes as input a set $\mathcal{D}$ of data and returns as output a decision, according to a set of conditions in the form (*attr op value*), with $op \in \{=, \neq, <, \leq, >, \geq\}$. It assumes the following form:

$$df(\mathcal{D}) = \begin{cases} decision1 & if \quad condition1 \\ decision2 & if \quad condition2 \\ decision3 & if \quad condition3 \end{cases}$$

This function calculates and tracks a decision on chain.

Each store process requires a different amount of activities to be executed on chain, which proportionally decrease the system performance. It also affects trustworthiness and privacy provided to the final user. We will discuss these aspects in the next sections.

7 TRUSTWORTHY DATA COLLECTION PROCESS

Our trustworthy data collection process is built around the concept of IoT Trust in Section 5 and data store in Section 6, and implements the architecture in Section 4. We propose three different implementations, varying on the basis of the considered validation rules.

7.1 Syntactic Verification

Figure 2 illustrates Syntactic Verification process. It implements a process where the trustworthiness of data points dp is checked against a set of syntactic rules $\mathcal{R}_I$. For each data point dp, smart contract SC_v in Definition 5.1 is executed and syntactic rules $\mathcal{R}_I$ evaluated against dp. Once the trustworthiness of a data point has been verified, the device gains access rights to the blockchain and a trustworthy data point dp (or data aggregation or decision) is stored in the blockchain using smart contract SC_s in Definition 6.1. We note that syntactic verification only considers syntactic requirements to identify trustworthy data, while it does not consider the expected sensor behavior.

Example 7.1. In a smart home, when the data coming from the sensors are submitted to the smart contract, syntactic thresholds are checked before storing the transactions. For instance, the threshold (*firmware_version*$\geq$2.0) is checked every time the syntactic verification is triggered.

7.2 Semantic Verification

Figure 3 illustrates Semantic Verification process. It implements a process where the trustworthiness of data points dp is checked against semantic rules $\mathcal{R}_D$, modeling the correct behavior of a specific

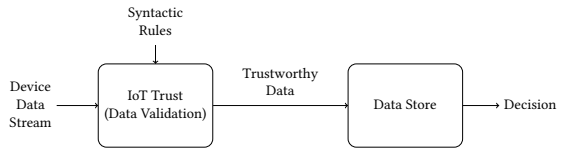


Fig. 2. Trustworthy Data Collection Process: Syntactic Verification

system/device. Semantic Verification is a 2-step process, consisting of phases setup and validation. Phase setup first builds semantic rules $\mathcal{R}_D$ as a behavioral model based on a set of data points *dp* used as training set. Phase validation provides these rules as input to IoT trust to evaluate the trustworthiness of data points *dp* and classify them on the basis of the level of compatibility with the behavioral model generated at phase setup. More in details, for each set of data points *dp*, smart contract SC_v in Definition 5.1 is executed and semantic rules $\mathcal{R}_D$ evaluated against them. Once the trustworthiness of data points has been verified, the device gains access rights to the blockchain and trustworthy data points *dp* (or data aggregations or decisions) are stored in the blockchain using smart contract SC_s in Definition 6.1.

We note that semantic verification models the expected device behavior as a means to verify trustworthy data. Of course, if training data contain inconsistencies due to syntactic violations, these inconsistencies will be modeled as well resulting in a decreased quality.

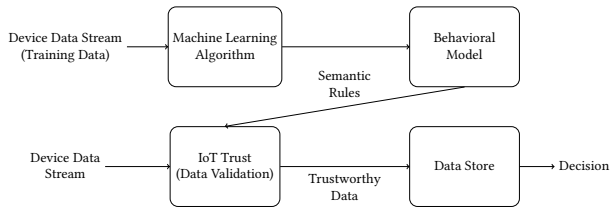


Fig. 3. Trustworthy Data Collection Process: Semantic Verification

Example 7.2. In the context of smart homes, the training data contain the behavior of a device under different conditions. For instance, the room temperature sensor can be monitored under different circumstances and retrieved data used as the ground truth. These data are used to build a model that represents the temperature sensor behavior during the day. Semantic verification uses this model to classify the newly coming readings. For instance, when a temperature reading arrives, the model is able to identify untrustworthy values (e.g., high room temperature in a winter day during working hours).

7.3 Hybrid Verification

Figure 4 illustrates Hybrid Verification process. It is a 2-step process, consisting of phases setup and validation, which integrates syntactic and semantic verification processes to increase data trustworthiness and the overall quality of our approach. Phase setup builds semantic rules $\mathcal{R}_D$ as a behavioral model, using a training set where data points *dp* have been successfully verified against syntactic rules. As a result, the created model only represents correct behavior filtering out inconsistencies due to syntactic violations. Phase validation provides these rules as input to IoT trust to evaluate the trustworthiness of data points *dp*. Data points are first filtered according to syntactic rules and then classified as trustworthy/untrustworthy on the basis of the level of

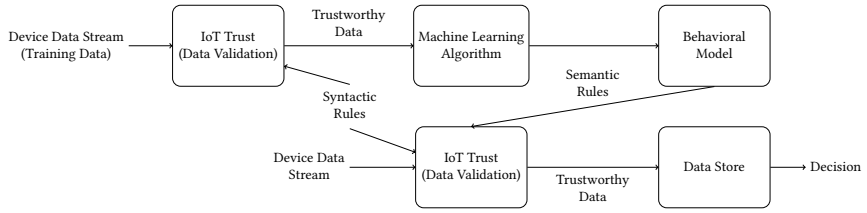


Fig. 4. Trustworthy Data Collection Process: Hybrid Verification

compatibility with the behavioral model generated at phase setup. More in details, for each set of data points *dp*, smart contract SC_v in Definition 5.1 is executed, and both syntactic rules $\mathcal{R}_I$ and semantic rules $\mathcal{R}_D$ evaluated against *dp*. Once the trustworthiness of data points has been verified, the device gains access rights to the blockchain and trustworthy data points *dp* (or data aggregations or decisions) are stored in the blockchain using smart contract SC_s in Definition 6.1.

7.4 Trustworthiness vs Privacy

The three implementation of our trustworthy data collection process permits to balance privacy and trustworthiness of our solution. Trustworthiness first depends on the amount of operations done on chain for data validation (IoT Trust) based on smart contract SC_v in Definition 5.1. Highest trustworthiness is achieved when both syntactic and semantic rules are evaluated on collected data (Hybrid Verification in Section 7.3); lower trustworthiness is achieved when either syntactic or semantic rules are evaluated (Syntactic Verification in Section 7.1 and Semantic Verification in Section 7.2, resp.). Trustworthiness and privacy then depend on the selected data store approach, that is, the amount of data points collected and stored in the blockchain by devices over time. To the aim of balancing trustworthiness and privacy, we defined a blockchain-integrated data store process that stores data on chain at different granularities: *i)* each single instance of collected data (data point), *ii)* an aggregation of a chunk of collected data (data aggregation), or *iii)* the final decision calculated based on collected data (decision). The three storing procedures, implemented as smart contract SC_s in Definition 6.1, increase visibility of the logic behind how the data are aggregated or a decision is made. At the same time, they address the privacy concerns guaranteeing access to different amount of data. For instance, while the owner of a smart home can access the full records of data points stored by sensors, the building manager can only access an aggregated version of the same data. In some cases, all the logic can be exported off chain and only the final decision stored on chain. We note that each of the options requires a different amount of activities to be executed on chain, which proportionally affect the system trustworthiness and privacy. In a nutshell, when single data points are stored in the blockchain, we achieve highest trustworthiness and lowest privacy, since all data are available for integrity checks and any decisions taken based on them can be replicated. When data aggregations are stored in the blockchain, privacy substantially increases, since it permits to hide details about the single data points, while trustworthiness decreases (decisions cannot be replicated). Finally, when decisions are stored in the blockchain, we achieve highest privacy and lowest trustworthiness, since all activities for taking a decision are done on chain, while only the decision is stored on chain.

8 PERFORMANCE EVALUATION

The adoption of a solution based on blockchain can backfire: management of a blockchain is costly, while a trustworthy data collection and verification process is resource demanding. For instance, auditing and reproducibility of the trustworthy data collection process require access to

all transactions history and could involve different participants (e.g., parties involved in generating the transactions). Performance of our solution mainly depends on two aspects: *i)* the amount of data collection and verification done on chain (smart contract SC_v in Definition 5.1) and *ii)* the amount/type of data stored on chain (smart contract SC_s in Definition 6.1). Considering point *i)*, the approach where syntactic and semantic rule validations are executed, insisting on the resources of the ledger, could be extremely demanding especially when the rate of collected measurements and the number of rules are high, introducing substantial latency. This approach provides high trustworthiness (see Section 7.4) but low performance.[4] Considering point *ii)*, when single data points are stored in the blockchain, highest performance is expected; when data aggregations are stored in the blockchain, a performance decrease is expected; when decisions are stored in the blockchain, lowest performance is expected, since all activities for taking a decision are done on chain. In the remaining of this section, we experimentally evaluate the performance of our solution varying data collection/verification and data store approaches.

8.1 Experimental Settings

The setup of the experiments consisted of 6 virtual machines running Ubuntu 16.4 deployed on top of CloudLab[5] data center. Each virtual machine was equipped with 20 vCPUs Intel Xeon E5-2640 v4 @ 2.40GHz and 64GB RAM and installed hyperledeger fabric. Hyperledger fabric is an open-source, permissioned ledger platform featuring smart contract (Chaincode) functionality. Hyperledger fabric employs channels and access control lists to manage controlled access and enable privacy among the network participants. It provides a modular and extendable architecture that permits plug in of different components to perform specific functions.

Contrary to the permissionless platforms such as Bitcoin and Ethereum, where all the nodes assume similar roles, fabric nodes have different roles (i.e., peers and orderers, endorsers), where the peers maintain copies of the ledger, orderers provide the communication channels and generate the new blocks (mining), and endorsers validate the submitted transactions. We configured a test network composed of 4 organizations, each one including two endorsing peers (they host ledgers and smart contracts), one orderer node used to generate the new blocks of data in the blockchain and one peer to keep a copy of the blockchain state. Each pair of organizations shared a channel with the total of 3 channels. Each channel specified relevant smart contracts (chaincode) to test all possible configurations of our solution as follows:

(1) Data Collection and Verification (Section 8.2): one smart contract validating syntactic rules and one smart contract validating semantic rules;
(2) Data Store (Section 8.3): one smart contract for data point store, one smart contract for data aggregation store, and one smart contract for decision store;
(3) Trustworthy Data Collection Process (Section 8.4): one smart contract for syntactic verification, one smart contract for semantic verification, and one smart contract for hybrid verification. These contracts provide the entire trustworthy data collection process as a sequential composition of contracts for data collection/verification and data store.

To assess the network performance, we relied on hyperledger caliper,[6] another project within the hyperledger family. Caliper benchmarks and measures the performance of different hyperledger blockchain networks including fabric. It providers an easy way to configure the network, smart contracts, channels, and workload. In our experiments, all the contracts have been implemented

[4]We note that the latency can be partially reduced by selecting a blockchain with high performance and fixed latency
[5]https://www.cloudlab.us
[6]https://www.hyperledger.org/projects/caliper

using Golang[7]. The results discussed in this section represent the average over 5 executions of the experiments.

8.2 Data Collection Verification

We first evaluated the performance of our network measuring the average latency and throughput retrieved by executing portions of our data collection/verification process (smart contract SC_v in Section 5) on chain, that is, syntactic rule validation and semantic rule validation. All tests were executed varying the transactions per second (tps) in 100, 200, 300, 400, 500 and the cardinality of syntactic rules in 10, 20, 30, 40, 50. We note that we are considering a challenging setup, as in many real scenarios the number of rules could be even less that 10. Semantic rule was implemented as a single classification model based on SVM classifier.

Every collected data point underwent a validation involving all syntactic rules, while semantic rule validation involved a single check on a batch of 50 data points. We note that since the semantic verification requires significant amounts of computational power and resources, our solution used the blockchain just to guarantee the integrity of the model by storing the hash of the model and the service endpoint. Figure 5(a) shows the latency when syntactic or semantic rules are validated. Concerning syntactic rule validation, the more the syntactic rules, the higher the performance decrease (higher latency, lower throughput). Also, a linear increase in latency was observed in the number of tps. Concerning semantic rule validation, the latency was low (<5s in the worst case) with respect to the latency of the syntactic rule validation (>24s in the worst case).

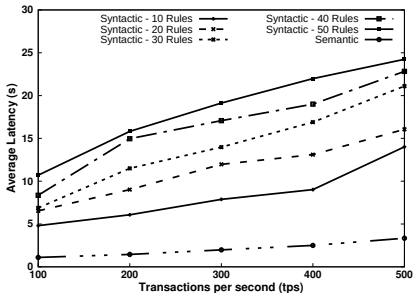
Figure 5(b) shows the throughput when 50 syntactic rules and one semantic rule are validated. The trend is similar to the one observed for the latency, where higher throughput (better performance) is retrieved for semantic rule validation: 12.5tps for semantic rule validation and 7.5tps for syntactic rule validation, in the worst case. This is due to the fact that semantic validation is usually based on a single rule consisting of a ML model, while syntactic validation considers a variety of syntactic rules (50 rules in our experiment) that affects the throughput.

8.3 Data Store

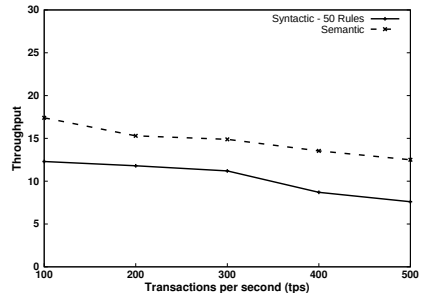
We then evaluated the performance of our network measuring the average latency and throughput retrieved by executing portions of our data store process (i.e., data point, data aggregation, decision store in smart contract SC_s in Section 6) on chain. All tests were executed varying tps in 100, 200, 300, 400, 500. Figure 5(c) show the latency when validating a single syntactic rule and varying the number of stored data points in 10, 30, 50. We note that the latency introduced by data point store (~2.5s in the worst case), still not negligible, is an order of magnitude less with respect to the latency introduced by syntactic rule validation in Figure 5(a) (~25sec in the worst case). Figure 5(d) shows similar results for the throughput, presenting a small decrease of less than 5tps when the number of validated data points is increased from 10 to 50.

Figure 5(e) and Figure 5(f) show the increase in latency and decrease in throughput when different parts of the data store process are executed on chain with 50 data points. We note that the overheads introduced by data aggregation and decision depend on the specific adopted function. In these experiments, we used standard functions for both, that is, average, min, max for data aggregation and a function based on threshold verification for decision. Data point store, still the most demanding in terms of performance, is the lowest in terms of latency increase and throughput decrease. Data aggregation and data decision are in fact sequential activities adding some latency on top of the data point store. In other words, the performance of data point store for latency and throughput (Figure 5(e) and Figure 5(f)) are increased by a fixed amount when data aggregation

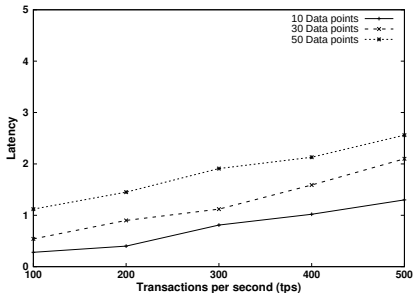
[7]https://golang.org/



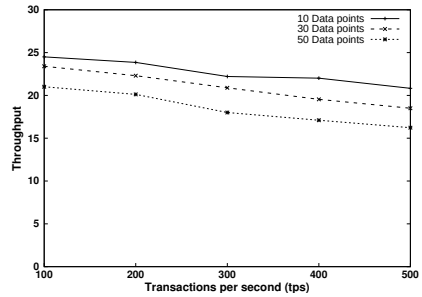
(a) Latency of syntactic and semantic rule validation



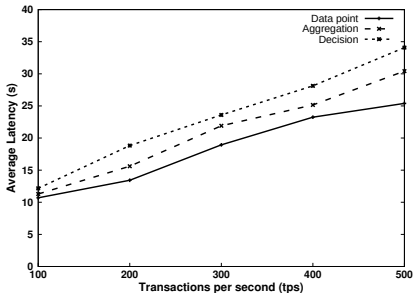
(b) Throughput of syntactic and semantic rule validation



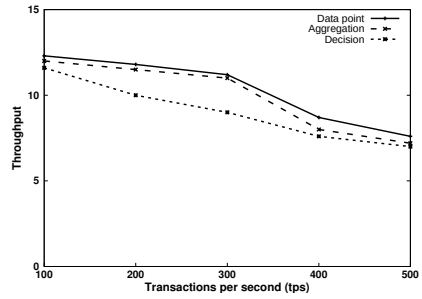
(c) Latency of data point store



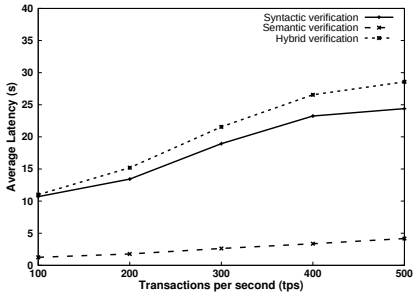
(d) Throughput of data points store



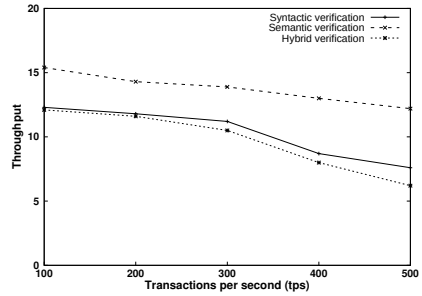
(e) Latency of data store



(f) Throughput of data store



(g) Latency of trustworthy data collection process



(h) Throughput of trustworthy data collection process

Fig. 5. Performance evaluation

and decision store are used. Data aggregation increases latency of 17% on average and decreases throughput of 5% on average, while data decision increases latency of 19% on average and decreases throughput of 12% on average.

8.4 Trustworthy Data Collection Process

Figure 5(g) and Figure 5(h) finally show the total latency and throughput for the three implementations of our trustworthy data collection process in Section 7. We consider a scenario with 50 syntactic rules and data point store, varying the transactions per second in 100, 200, 300, 400, 500. Comparing semantic and syntactic verification, as expected, semantic verification introduced the lowest latency (<5s in the worst case) and achieved the highest throughput (~12tps in the worst case), while syntactic verification introduced the highest latency (~25s in the worst case) and achieved the lowest throughput (between 7tps and 8tps in the worst case). Hybrid verification summed up the latency in syntactic and semantic verification, while decreased the throughput to less than 6tps in the worst case.

To conclude, we note that different configurations, which have an impact on the provided level of trustworthiness (see Section 7.4), introduce a not-negligible cost in terms of performance. This suggests the importance of selecting the proper level of trustworthiness for the domain of interest to keep the overall performance under control.

9 QUALITY EVALUATION

We experimentally evaluated the quality of our trustworthy data collection process in two scenarios, which cover the entire spectrum of our approach. Scenario 1 "Smart Home Monitoring" considers data collected from more than one hundred sensors in two smart homes. It is a simulated scenario testing the trustworthy data collection process using semantic and hybrid verifications. It generates the ML model of the smart home behavior, and simulates streams of sensor data points whose trustworthiness needs to be verified. Scenario 2 "Mobile Network Roaming" considers data collected from the switching infrastructure of the ETISALAT Mobile Network. It is a real-world scenario testing the trustworthy data collection process using syntactic and hybrid verification, and real streams of data describing roamers traffic on the mobile network. It also evaluates how a decision process for high-usage warning can improve in quality when data filtered by our approach are used.

9.1 Scenario 1: Smart Home Monitoring

Smart Home Monitoring uses data for activity recognition in the data set *Home Setting Using Simple and Ubiquitous Sensors*.[8] The data set contains data collected by two smart homes using a partially overlapped sets of 77 sensors and 84 sensors, respectively. It aimed to verify the quality of our trustworthy data collection process when filtering those untrustworthy sensor data that might hamper a sound decision process.

Our evaluation process was composed of the following steps: smart contract definition, syntactic rule validation, and semantic rule validation. We note that we both executed semantic verification in Section 7.2 and hybrid verification in Section 7.3. Semantic verification applied semantic rule validation in isolation, while hybrid verification applied syntactic and semantic rule validation in a sequence, both followed by the data point store in Section 6. For conciseness, in the following, we focused on syntactic and semantic rule validation only.

9.1.1 Smart Contract Definition. We defined two smart contracts for syntactic and semantic rule validation (see the execution flow in Figure 1). Each contract was composed of two parts: *i)* main functions modeling rule validation, *ii)* utility functions reducing the code complexity and improving performance. A total of five functions have been implemented.

- **Main Functions**

[8]Data set available at https://courses.media.mit.edu/2004fall/mas622j/04.projects/home/.

  - *SyntacticValidation*: it takes as input a data point and the device signature; it then loads the rules relevant for the specific device type. Only if the data point satisfies all the rules a positive response is returned (see Section 5.2.1).
  - *SemanticValidation*: it takes as input a data point and loads the ML model that has been set using function *SetModel*. Function *SemanticValidation* then calls an external REST API to perform the verification off chain (see Section 5.2.2).
- **Utility Functions** includes *AddRules* to insert new rules for *SyntacticValidation*, *SetModel* to update the ML model that is used for *SemanticValidation*, *AddDevice* to register a new device signature in the blockchain.

9.1.2 Syntactic Rule Validation. We manually prepared and cleaned data for our experiments retrieving a final data set containing 503 data points collected from 91 unique sensors as follows. One-hot encoding was first performed to properly encode and represent the features of the data set. In addition, to simulate the quality of the sensor data, we labeled rare data points as untrustworthy due to incorrect sensor measurements (i.e., *syntactic* errors resulting in syntactic rule violations).[9] The rest of the data points were labeled as trustworthy. Accordingly, the data set contained 360 trustworthy data points (sensor measurements) and 143 untrustworthy data points that, where possible, were manually corrected according to syntactic rules. These activities resulted in a data set used as the baseline of our experiments with 468 trustworthy data points and only 35 untrustworthy data points. Starting from this data set, we evaluated the performance of semantic verification (using all 503 data points) and hybrid verification (using only the 468 trustworthy data points) in Section 7. The reason why syntactic verification was not tested alone is that its quality highly depended on the amount of syntactic errors in the data set and could then be biased by how the data set was built.

9.1.3 Semantic Rule Validation. The quality evaluation passed through the definition of the behavioral model representing the semantic rule in Section 7. To this aim, we executed an extensive analysis that we conducted to better understand the data and its characteristics as follows.

- We first identified those features (sensors) that could contain more information in the target data set. This was done by building a Random Forest model using Scikit-learn implementation, a feature engineering tool. In building this model, we set the number of trees to 1000 and the model criteria to the Gini impurity. According to our analysis, the most valuable features in predicting the trustworthiness of a sensor measurement were mainly related to the time of the activity (i.e., starting time) and the length of the performed activity (i.e., how long the activity goes on). In addition, different sensors also contributed to the quality of prediction. The top-most informative sensors were sensor number 74 (i.e., Kitchen, Refrigerator), 141 (i.e., Living room, TV), and 115 (i.e., Kitchen, Microwave).
- We then evaluated the correlation between different features (sensors) and, more importantly, with the target class labels (i.e., trustworthy/untrustworthy), using *Pearson Correlation* (NumPy implementation) with its Coefficient ranging from -1 (showing negative correlation) to +1 (showing positive correlation). Overall, our analysis showed that a large number of features had considerable correlation. For example, sensors 144 (i.e., Kitchen, Refrigerator) and 145 (i.e., Kitchen, Cereal) were highly and positively correlated with the trustworthiness of the data. On the other hand, sensors 133 (i.e., Kitchen, Cabinet) and 135 (i.e., Kitchen, Drawer) were highly but negatively correlated with the trustworthiness of the data. In addition to the sensors, the execution window of an activity presented a high and positive correlation with the class labels. Hence, in addition to the sensor features, the time-related features could be

[9]We note that this assumption does not apply in all scenarios and rare points can be managed using unsupervised approaches.

possible sources of information to predict the validity of the measurements within the IoT environments.

- We then used Principal Component Analysis-PCA (Scikit-learn implementation) to reduce the dimensions and sparsity of our data set, retrieving a smaller data set with very representative dimensions. The retrieved results showed that data clearly formed two groups. This was probably due to the fact that the original data contained activities of two people in two different smart homes. Moreover, we noticed that the majority of untrustworthy sensor measurements were within one of the two groups. Since all untrustworthy sensor measurements originated from rare sensors with lowest number of activations, they can correlate with the frequency of activation. Overall, our results can indicate that the untrustworthy measurements were mainly obtained from one of the smart homes and from sensors with less frequency of activation.
- We applied Elbow method to retrieve an indication of the best number of clusters modeling our data set. Our results showed that the optimal number of clusters could be within 10 and 20, where the distortion (clustering error) had the largest drop and then became (more or less) stable. We also performed clustering based on the identified number of clusters. Each cluster can represent part of the data with considerable similarity. The results of the clustering could help in identifying the most common trustworthy or untrustworthy sensor measurements (these are basically the centroids of the clusters).

After this analysis, we used a portion of our data set to build predictive models (classifiers) capable of performing classification and predicting the trustworthiness of a measurement. To this aim, we applied *k-fold cross validation* evaluation methodology, with k=10 and k=20 (the optimal number of clusters boundaries). We note that since we obtained similar results with k=20, for conciseness, we only present the results for k=10.

- The data set was randomly split into 10 non-overlapping subsets, where in every iteration, $\frac{9}{10}$ of data points were used to train the prediction model (classifier) and the rest $\frac{1}{10}$ was used to test it.
- The quality of the prediction was measured in terms of popular metrics, namely, *accuracy*, *precision*, and *recall* [25].

It is important to recall that our experiments aim to evaluate the need of an architectural pattern (Section 3) for filtering out untrustworthy data (sources) due to unintentional sensor malfunctioning and failures, and the quality of corresponding trustworthy data collection process (Section 7). To this aim, we adopted standard classification models (i.e., Support Vector Machines (SVM), Logistic Regressions, K-Nearest Neighbors, Random Forest and Gradient Boosting) with no optimization, to test whether the untrustworthiness identification problem can be reduced to a standard classification problem. With these models, we have run the experiments with the following model parameters: support vector machines with Radial Basis Function (RBF) kernel, gamma factor set to 0.10 and regularization parameter set to 10.0. With gradient boosting, we chose *deviance* as for the choice of loss function to be optimized. With logistic regressions, we used L2 norm for the regularization and limited-memory BFGS solver for the optimization algorithm. With random forest, we set the number of trees in the forest to 10. Finally, with K-Nearest Neighbors, we set the number of neighbors to 10. We trained these models on each training set, and evaluated them on each test set. The models predicted the trustworthiness of a measurement within the test set and the predictions were compared against the human-annotated class labels (i.e., ground truth).

From the results in Table 1, we observe an increase in the quality of the data after both semantic and hybrid verification, with substantially higher quality when hybrid verification is used. Hybrid verification, in fact, used the syntactically correct data set composed of 468 trustworthy records

Table 1. Quality results

Trustworthy Data Collection	Prediction Model	Accuracy	Precision	Recall
Semantic Verification	Logistic Regression	0.29	**0.62**	0.65
	K-Nearest Neighbors	0.38	0.59	**0.66**
	Support Vector Machines	**0.46**	0.55	0.60
	Random Forest	0.31	0.60	0.63
	Gradient Boosting	0.30	0.61	0.64
Hybrid Verification	Logistic Regression	0.48	**0.84**	0.88
	K-Nearest Neighbors	0.77	0.83	**0.91**
	Support Vector Machines	**0.86**	0.83	0.89
	Random Forest	0.67	0.83	0.90
	Gradient Boosting	0.63	0.83	0.89

for building the ML model used in semantic rule validation. We also observe that the increase in data quality is independent from the prediction model. For accuracy metric, the best results were achieved by support vector machines. The accuracy of this model reached 0.86 when the syntactic errors were filtered out using syntactic rules (hybrid verification in Section 7); it reached 0.46 when the syntactic errors were not filtered (semantic verification in Section 7). All other classifiers performed inferior to this classifier in terms of accuracy, but still clearly showing a substantial increase in the quality when hybrid verification in Section 7 was used.

In terms of precision, all prediction techniques performed well and achieved relatively similar results. Still, the best results were obtained by logistic regression model with the precision of 0.84, for hybrid verification, and 0.62, for semantic verification. In terms of recall, *K*-Nearest Neighbors technique marginally outperformed the other algorithms by reaching a recall of 0.91 for hybrid verification and 0.66 for semantic verification.

Our results show the potential of machine learning techniques as an important means for increasing the trustworthiness of data collection and verification with their prediction quality and performance improvement. Our results also prove that the problem of identifying untrustworthy data (sources) can be attacked as a standard classification problem, especially when the cause of untrustworthiness is a sensor malfunctioning or a configuration error. This is extremely different from existing solutions (e.g., [12, 26]) that often focus on poisoning attacks carried out by active adversaries and employ complex deep learning techniques to separate trustworthy and untrustworthy data sources.

9.2 Scenario 2: Mobile Network Roaming

Mobile Network Roaming used a data set coming from ETISALAT mobile network consisting of a Transferred Account Procedure (TAP) file. TAP is the mechanism supported by wireless operators in the exchange of roaming billing information. It is distributed and involves multiple networks including the home network of and external networks visited by the users. TAP files include usage profiles in the form of call detailed records (CDRs), generated for mobile calls/service requests and contain information on subscriber identification, call charging, and obtained services (call/Internet browsing). When a user (roamer) visits another mobile network (visited network) and accesses its services a CDR is created. These records are transferred to the billing system for rating and pricing. TAP files are continuously sent to roaming partners on a regular basis (TAP-out process), as frequently as possible to enable monitoring of high usage and possible fraud.

Today, a contractual agreement between a user and its home network provider requires to notify the user when a high-usage roaming profile is identified and the trend is towards exceeding a specific threshold value. This notification is under the responsibility of the visited network, which adopts the same threshold agreed by the home network provider and the user. The more the

Table 2. CDR features of Interest

Feature	Description
RECORD_TYPE	Record type
CALL_REFERENCE	Unique identifier for the call
CALL_NO	The number of the callees
TOTAL_DURATION	Activity duration
CHARGED_UNITS	How many units have been charged for a specific call
MSISDN	A number used to identify a mobile phone number internationally (caller number)
CALLEVENT_START_TIMESTAMP	When the conversation started

accuracy and precision of this notification system, the higher the revenue for the home and visited network operators, on one side, the lower the risk of incurring in penalties for missing contractual agreements, on the other side. CDRs in TAP files are therefore continuously analyzed to show the roamer usage trends; in particular, the derivative of the usage is used to predict when the high-usage threshold value will be exceeded according to the observed trend. This prediction is however affected by untrustworthy data: TAP files might contain CDRs with wrong data (measurement error), wrong service (error in the identification of the used service), inconsistent data due to network errors or malicious users wanting to hide their real behavior.

Scenario 2 aimed to verify the accuracy of our approach when filtering untrustworthy CDRs in TAP files and its impact on the final decision process (high-usage warning for Internet service only). Our evaluation process considered a TAP file from the ETISALAT network consisting of 43,521 CDRs from 14 users monitored over a period of one week (September 30, 2019 – October 6, 2019). It is composed of the following activities: smart contract definition, data preparation, syntactic rule validation, semantic rule validation, decision accuracy. We note that we both executed syntactic verification in Section 7.1 and hybrid verification in Section 7.3. Syntactic verification applied syntactic rule validation in isolation, while hybrid verification applied syntactic and semantic rule validation in a sequence, followed by decision store in Section 6.

9.2.1 Smart Contract Definition. We defined three smart contracts for syntactic and semantic rule validation based on CDR features in Table 2. The first smart contract defined two syntactic rules (i.e., *PingSequence* and *IsValidPing*) for syntactic rule validation, the second smart contract defines a semantic rule (i.e., *AbnormalActivity*) for semantic rule validation, the third contract a function (i.e., *SendWarning*) for decision store, as follows.

- *PingSequence*: it takes a set of CDRs as input and summarizes n consecutive ping messages in a single one (the one with the most recent timestamp). Ping messages are used for network management and should not be considered in the high-usage monitoring process. They have 0 as the call number (CALL_NO) and 0 as the total charged prize (CHARGED_UNITS).
- *IsValidPing*: it takes a CRD as input and returns a Boolean value stating whether it is a valid ping message as output, that is, it contributes to filter out those ping messages with CHARGED_UNITS higher than 0.
- *AbnormalActivity*: it takes a CRD as input and returns a Boolean value stating whether it is a valid CDR as output, that is, it contributes to filter out those CDRs with a cost per unit of time (CHARGED_UNITS/TOTAL_DURATION) that is much lower/higher than the average cost per unit of time.
- *SendWarning*: it takes a CRD, the high-usage threshold H of the roamer, the time threshold T used for triggering a high-usage warning message as input, and returns a Boolean value stating whether a warning should be sent as output, according to the trend modeled by the first derivative calculation for CDRs of type Internet service.

Table 3. Filtering results

(a) Syntactic Rule Validation

			Automatic Filter			
Day	# of subsets	Avg. # of errors	3	5	7	9
September 30, 2019	12	14	63,40%	83,13%	88,57%	92,86%
October 1, 2019	12	20	60,10%	82,11%	90,01%	91,11%
October 2, 2019	12	18	56,20%	85,91%	87,75%	97,71%
October 3, 2019	12	23	55,21%	80,09%	85,32%	92,86%
October 4, 2019	12	21	60,07%	84,19%	87,12%	93,81%
October 5, 2019	12	18	65,11%	86,32%	89,41%	94,50%
October 6, 2019	12	21	55%	74,31%	98,21%	90,90%
Average	12	19,28	59,30%	82,30%	89,49%	93,40%

(b) Semantic Rule Validation

Day	Precision	Recall
September 30, 2019	97,54%	70,08%
October 1, 2019	96,28%	64,71%
October 2, 2019	92,38%	73,31%
October 3, 2019	95,18%	72,87%
October 4, 2019	93,46%	65,95%
October 5, 2019	95,35%	68,59%
October 6, 2019	94,85%	70,81%
Average	95,00%	69,47%

9.2.2 Data Preparation. The first step of our process consisted in the preparation of the TAP file retrieved from the ETISALAT network for data analysis. First, all features of interest in Table 2 were selected and CDRs ordered according to MSISDN and CALLEVENT_START_TIMESTAMP. Then, starting from the *original data set*, we generated two additional data sets. The first data set (*manual data set*) was generated by manually filtering the CDRs in the original data set. To this aim, we classified each CDR according to its type, namely call records, Internet records, ping records, and only maintained the Internet records. Internet records with abnormal cost per unit of time was then filtered out. Finally, *PingSequence* and *IsValidPing* were manually and recursively applied to remove all consecutive and invalid ping messages sent by a mobile device to the visited network. We note that the *manual data set* represented our baseline with 100% accuracy. The second data set (*filtered data set*) was generated by automatically filtering errors according to our syntactic and semantic rule validation as presented in Sections 9.2.3 and 9.2.4.

9.2.3 Syntactic Rule Validation. The data in the original data set contained syntactic errors that could affect the quality of the high-usage warning process. To this aim, we applied our syntactic validation in Section 7.1 to filter them out and increase trustworthiness of collected CDRs. We then applied functions *PingSequence* and *IsValidPing* in Section 9.2.1 to filter ping messages, not contributing to the total amount of charged units used by the high-usage warning process, as follows: *i)* we removed all invalid pings, that is, ping messages with CHARGED_UNITS greater than zero, using function *IsValidPing*; *ii)* we summarized n consecutive valid pings, that is, messages with no CALL_NO and CHARGED_UNITS equals to 0, with one ping (i.e., the one with the most recent timestamp) using function *PingSequence*.[10] The main difference between automatic and manual filtering is that manual filtering recursively applied function *PingSequence* achieving a higher filtering accuracy.

We computed the accuracy of our syntactic rule validation, by comparing the number of erroneous pings that remained unfiltered in the filtered data set. Table 3(a) shows our results, discussing the accuracy of our syntactic verification day-by-day and on average. For each day, we present the accuracy of our automatic filtering considering all 14 users. The last row presents the average on the whole week. It is important to note that the accuracy of our filtering process increases as the number n of consecutive pings to be summarized increases. This is due to the fact that our syntactic rule *PingSequence* is not recursive and for small n leaves a not-negligible numbers of consecutive pings in the data set. On the other side, having a value of n too big might bring to the opposite scenario, where less than n consecutive pings are not even considered for filtering. According to our data, the optimum choice was n=9, where 93.4% of the ping errors were filtered out.

[10]We note that consecutive pings can be created due to cell changes and network delays.

9.2.4 Semantic Rule Validation. The data filtered according to syntactic rule validation were checked against a semantic rule aiming to remove those CDRs whose type was not Internet service. We then applied function *AbnormalActivity* in Section 9.2.1 to filter CDRs as follows. First, we calculated the average cost per unit of time and standard deviation of Internet records. Then, we filtered out all CDRs above the 95th percentile.

We computed the accuracy and precision of our algorithm in filtering CDRs, by observing *i)* the number of CDRs of type Internet service with abnormal cost per unit of time and the number of type call that remained unfiltered in the filtered data set, *ii)* the number of CDRs of type Internet service that were filtered out. Table 3(b) shows our results discussing the accuracy and precision of our semantic rule validation day-by-day and on average. For each day, we present the quality of our filtering considering all 14 users. The last row presents the average on the whole week. We note that our aim was to maximize the accuracy of our algorithm (95% on average), minimizing the number of unfiltered CDRs at point *i)*, at the price of a decrease precision (69,47% on average) due to a not-negligible amount of filtered CDRs at point *ii)*. The reason driving our choice was to test our decision quality in a challenging scenario where a substantial amount of correct data were filtered out, while a negligible amount of incorrect data remained in the data set.

9.2.5 Decision Accuracy. We used the three data sets produced in the previous steps to evaluate the effects our approach had on the quality of the high-usage warning process for Internet service: *i)* the original TAP file including all syntactic errors (original data set), *ii)* the filtered TAP file, that is, the noisy TAP file with untrustworthy CDRs filtered according to our approach (filtered data set), *iii)* the manually filtered TAP file (manual data set). The TAP files were used to identify the cumulative distribution of network usage (CHARGED_UNITS) over time. We considered the notification process of roamer high usage as our decision process and evaluated its quality. The decision process, inspired by ETISALAT, was a simplified approach based on first derivative calculation; this choice was done to reduce the impact of the selected algorithm on the quality of our results. The high-usage warning process worked as follows.

(1) The three data sets were partitioned in 98 data sets each, where each data set referred to a specific user (MSISDN) and day of observation (CALLEVENT_START_TIMESTAMP). The high-usage threshold H and the time threshold T were also defined. The cumulative distribution of network usage over time was calculated per user, per day.
(2) Upon selecting a specific data set, the first derivative $f'(t)$ at time t was calculated as the secant passing through two consecutive points (x_t,y_t) and (x_{t+1},y_{t+1}).
(3) The predicted time $\bar{t}$ needed to reach the threshold H was calculated by solving the system of two equations including the high-usage threshold $y=H$ and the straight line $y=ax+b$ modeling $f'(t)$. The solution (x_p,y_p) was such that $\bar{t}=x_p-t$.
(4) If $\bar{t}<T$, a warning was triggered.
(5) If no warning was triggered the same process was repeated from Step 2 with $t=t+2$. Otherwise, if a warning was triggered or the data set completed with no warning, the next data set (if any) was analyzed starting from Step 2 and $t=0$.

We executed the above algorithm on our three data sets, fixing high-usage threshold $H=30$ charged units and time threshold $T=30$ min. We then compared the average difference in warning time using the data sets and taking the manual data set as our benchmark. For all data sets, the same number of warnings was triggered. However, the error distance in minutes substantially varied. Using the original data set, we triggered the high-usage warnings 79 minutes in advance on average with respect to the one retrieved with the manual data set. This was due to the fact that the original data set contained many unfiltered CDRs with high cost per unit of time (CHARGED_UNITS/TOTAL_DURATION). On the other hand, the filtered data set while removing

the majority of the outliers, also erroneously removed part of CDRs of type Internet service. We recall that this was due to the choice of testing our approach in a suboptimal and challenging scenario. We then observed a delay of 18 min on average in the forwarding of the high-usage warning, clearly outperforming the performance of the original data set. We also observe that no warning notification happened after the threshold was breached.

To conclude, it is very important to underline that our approach is prone to optimization, since all parameters of the decision are stored on chain and can be tuned over time, according to the performance observed. Better tuning the percentile in Section 9.2.4 for semantic rule validation, and the two thresholds H and T used for decision, as well as smarter ways of calculating the first derivative, can bring to better results and performance. The overall trustworthiness, instead, is not affected by optimization, since all steps of our analysis are stored on chain and can then be used for assurance verification.

10 CONCLUSIONS

In this paper, we presented a methodology for collecting trustworthy data that complements existing trustworthy decision processes. The intuition behind our approach is that the more trustworthy data, the higher the decision quality. Our methodology for trustworthy data collection in IoT environments filters out untrustworthy data coming from smart devices on the basis of either syntactic rules and semantics rules. It is based on blockchain and smart contracts supporting traceability of decisions taken according to collected data and generated behavioral model. This approach has been also referred to as open execution in literature [23], since it provides full visibility of the system execution. We also specified different implementations of the trustworthy data collection process with varying trustworthiness and privacy, which have been experimentally evaluated in a real-world scenario using Hyperledeger fabric blockchain. The solution in this paper leaves space for future work. First, we will investigate formal properties of our approach with particular reference to properties trustworthiness and privacy. Then, we will investigate the impact of standard classification algorithms on the performance of semantic rule validation. We will investigate how the approach in this paper can be extended to consider complex scenarios where different processes and systems are composed. Finally, we will consider penalty-based unsupervised machine learning techniques when dealing with application-specific verticals.

ACKNOWLEDGMENTS

Research supported, in parts, by EC H2020 Project CONCORDIA GA 830927 and Università degli Studi di Milano under the program “Piano sostegno alla ricerca 2019”.

REFERENCES

[1] M. Ambrosin, M. Conti, A. Ibrahim, A. Sadeghi, and M. Schunter. 2018. SCIoT: A Secure and sCalable End-to-End Management Framework for IoT Devices. In *Proc. of ESORICS 2018*. Barcelona, Spain.

[2] C.A. Ardagna, R. Asal, E. Damiani, C. Pahl, N. El Ioini, and C. Pahl. 2019. Trustworty IoT: An Evidence Collection Approach based on Smart Contracts. In *Proc. of SCC 2019*. Milan, Italy.

[3] C.A. Ardagna, R. Asal, E. Damiani, and Q.H. Vu. 2015. From Security to Assurance in the Cloud: A Survey. *ACM CSUR* 48, 1 (August 2015), 2:1–2:50.

[4] C.A. Ardagna, E. Damiani, J. Schutte, and P. Stephanow. 2017. A Case for IoT Security Assurance. In *Internet of Everything*, B. Di Martino, K. Ching Li, L. Yang, and A. Esposito (Eds.). Springer.

[5] S. Bonomi, M. Casini, and C. Ciccotelli. 2018. B-CoC: A Blockchain-based Chain of Custody for Evidences Management in Digital Forensics. *arXiv preprint arXiv:1807.10359* (2018).

[6] E. Damiani T. Dimitrakos N. El Ioini C. Pahl C. Ardagna, R. Asal. 2018. Certification-Based Cloud Adaptation. *IEEE Transactions on Services Computing* PP (01 2018). https://doi.org/10.1109/TSC.2018.2793268

[7] C. Dai, D. Lin, E. Bertino, and M. Kantarcioglu. 2008. An approach to evaluate data trustworthiness based on data provenance. In *Proc. of SDM 2008*. Auckland, New Zealand.

[8] Ericsson. 2016. Bootstrapping Security - The Key to Internet of Things Access Authentication and Data Integrity. (February 2016). Ericsson white paper, 284 23-3284, http://www.ericsson.com/res/docs/whitepapers/wp-iot-security.pdf.

[9] D. Gonzales, J. M. Kaplan, E. Saltzman, Z. Winkelman, and D. Woods. 2017. Cloud-Trust, a Security Assessment Model for Infrastructure as a Service (IaaS) Clouds. *IEEE TCC* 5, 3 (July 2017), 523–536.

[10] M. Irfan, H. Abbas, Y. Sun, A. Sajid, and M. Pasha. 2016. A framework for cloud forensics evidence collection and analysis using security information and event management. *Security and Communication Networks* 9, 16 (2016), 3790–3807.

[11] U. Jayasinghe, A. Otebolaku, T.-W. Um, and G.M. Lee. 2017. Data centric trust evaluation and prediction framework for IoT. In *Proc. of ITU K 2017*. Nanjing, China.

[12] S. Kariyappa and M.K. Qureshi. 2019. Improving adversarial robustness of ensembles with diversity training. In *arXiv:1901.09981*.

[13] N. Karthik and V.S. Ananthanarayana. 2017. Data trust model for event detection in wireless sensor networks using data correlation techniques. In *Proc. of ICSCN 2017*. Chennai, India.

[14] L. Li and A. Ghasemi. 2019. IoT-Enabled Machine Learning for an Algorithmic Spectrum Decision Process. *IEEE Internet of Things Journal* 6, 2 (April 2019), 1911–1919.

[15] B. Liu, X.L. Yu, S. Chen, X. Xu, and L. Zhu. 2017. Blockchain based data integrity service framework for IoT data. In *Proc. of ICWS 2017*. Honolulu, HI, USA.

[16] Auqib Hamid Lone and R. N. Mir. 2019. Forensic-chain: Blockchain based digital forensics chain of custody with PoC in Hyperledger Composer. *Digital Investigation* 28 (2019), 44–55.

[17] B.K. Mohanta, S.S. Panda, U. Satapathy, D. Jena, and D. Gountia. 2019. Trustworthy Management in Decentralized IoT Application using Blockchain. In *Proc. of ICCCNT 2019*. Kanpur, India.

[18] K.R. Özyılmaz and A. Yurdakul. 2017. Work-in-progress: integrating low-power IoT devices to a blockchain-based infrastructure. In *Proc. of EMSOFT 2017*. Seoul, South Korea.

[19] J.H. Park, J.Y. Park, and E.N. Huh. 2017. BLOCK CHAIN BASED DATA LOGGING AND INTEGRITY MANAGEMENT SYSTEM FOR CLOUD FORENSICS. *Computer Science & Information Technology* (2017), 149–159.

[20] R. Rani, S. Kumar, and U. Dohare. 2019. Trust Evaluation for Light Weight Security in Sensor Enabled Internet of Things: Game Theory Oriented Approach. *IEEE Internet of Things Journal* 6 (2019), 8421–8432.

[21] M. Raya, P. Papadimitratos, V.D. Gligor, and J.-P. Hubaux. 2008. On data-centric trust establishment in ephemeral ad hoc networks. In *Proc. of INFOCOM 2008*. Phoenix, AZ, USA.

[22] J. Rowley. 2007. The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science* 33, 2 (2007), 163–180.

[23] K. Salah, E. Damiani, A. Al-Fuqaha, T. Martin, K. Taha, and M.K. Khan. 2018. Open Execution – The Blockchain Model. *IEEE Blockchain Technical Briefs* (December 2018).

[24] H. Sato, A. Kanai, S. Tanimoto, and T. Kobayashi. 2016. Establishing Trust in the Emerging Era of IoT. In *Proc. of IEEE SOSE 2016*. Oxford, UK.

[25] M. Schedl, H. Zamani, C. Chen, Y. Deldjoo, and M. Elahi. 2018. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval* 7, 2 (01 Jun 2018), 95–116. https://doi.org/10.1007/s13735-018-0154-2

[26] S. Shen, S. Tople, and P. Saxena. 2016. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proc. of ACSAC 2016*. Los Angeles, CA, USA.

[27] A. Sheth. 2016. Internet of Things to Smart IoT Through Semantic, Cognitive, and Perceptual Computing. *IEEE Intelligent Systems* 31, 2 (March-April 2016), 108–112.

[28] P. Shi, H. Wang, S. Yang, C. Chen, and W. Yang. 2019. Blockchain-based trusted data sharing among trusted stakeholders in IoT. *Software: Practice and Experience* (August 2019), 1–14. https://doi.org/10.1002/spe.2739

[29] M. Sigwart, M. Borkowski, M. Peise, S. Schulte, and S. Tai. 2019. Blockchain-based data provenance for the internet of things. In *Proc. of IoT 2019*. Bilbao, Spain.

[30] S. Suhail, C.S. Hong, M.A. Lodhi, F. Zafar, A. Khan, and F. Bashir. 2018. Data trustworthiness in iot. In *Proc. of ICOIN 2018*. Chiang Mai, Thailand.

[31] Salman Taherizadeh, Andrew C. Jones, Ian Taylor, Zhiming Zhao, and Vlado Stankovski. 2018. Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review. *Journal of Systems and Software* 136 (2018), 19 – 38. https://doi.org/10.1016/j.jss.2017.10.033

[32] F. Xia, L.T. Yang, L. Wang, and A. Vinel. 2012. Internet of Things. *International Journal of Communication Systems* 25 (September 2012), 1101–1102. Issue 9.

[33] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao. 2017. A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet of Things Journal* 4, 5 (October 2017), 1250–1258.

[34] Z. Yang, K. Yang, L. Lei, K. Zheng, and V.C.M. Leung. 2018. Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet of Things Journal* 6, 2 (2018), 1495–1505.
[35] I. Yen, F. Bastani, N. Solanki, Y. Huang, and H. San-Yih. 2018. Trustworthy Computing in the Dynamic IoT Cloud. In *Proc. of IRI 2018*. Salt Lake City, UT, USA.