Deliverable D2.1 Joint state of the art on detection technologies for future networks

Daniele Antonioli, Christophe Gaston, Houda Jmila, Chana Weil-Kennedy

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

anr agence nationale de la recherche

FRANCE 2030 PROGRAMME DE RECHERCHE RÉSEAUX DU FUTUR

Grant agreement ANR-22- PEFT-0009

Deliverable D2.1
Joint state of the art on detection technologies for future networks

Delivery date	30/04/2025
Version	1.0
Editor	Houda Jmila
Authors	Daniele Antonioli, Chritophe Gaston, Houda Jmila, Chana Weil-Kennedy
Dissemination	Public
Keywords	Smart Home, Matter, Runtime Verification, Distributed Systems, IA, IDS

History

Version	Date	Modification	Authors
1.0	30/04/2025	Final version	DA, CG, HJ, C W-K

Executive summary

The evolution of network technologies including the rise of 5G, 6G, and the Internet Of Things (IoT) is fundamentally transforming the landscape of digital connectivity. These new-generation networks introduce unprecedented levels of complexity, distribution, and dynamism, making the detection of anomalies a critical challenge for ensuring security, reliability, and performance.

This report provides a state-of-the-art survey of anomaly detection techniques tailored for future networks. It focuses on two main technical approaches: Artificial Intelligence (AI)-based Intrusion Detection Systems (IDSs) and Runtime Verification (RV), a formal method for monitoring system behavior. In addition to exploring these methodologies, the report presents a real-world application: anomaly detection in smart home networks, with a focus on the Matter protocol.

Key findings of the report highlight the need for adaptive traffic representation, the combination of AI and formal methods, and the need to develop efficient detection technologies for real-world applications like smart homes. The report concludes with recommendations for future research directions to build scalable, efficient, and robust anomaly detection systems for next-generation networks.

Table of content

List of Acronyms

ML Machine Learning

AI Artificial Intelligence

HMM Hidden Markov Model

RV Runtime Verification

BLE Bluetooth Low Energy

BTP Bluetooth Transport Protocol

IoT Internet Of Things

PAKE Password-Authenticated Key Exchange

PASE Passcode Based Session Establishment

CASE Certificate Based Session Establishment

SIGMA SIGn-and-MAc

CA Certificate Authority

mDNS Multicast DNS

DNS-SD DNS-Based Service Discovery

CHIP Connected Home IP

CSA Connectivty Standards Alliance

SDK Software Development Kit

MitM Machine-in-the-Middle

TLS Transport Layer Security

MRP Message Reliability Protocol

MUD Manufacturer Usage Description

MitM Machine-in-the-Middle

IDS Intrusion Detection System

TLS Transport Layer Security

1 Introduction

1.1 Context and Objectives

The rapid evolution of network technologies and the increasing complexity of network infrastructures have brought forth new challenges in ensuring the security and reliability of future networks, including 5G/6G and IoT ones. As we move towards the widespread adoption of future networks, the need for effective anomaly detection mechanisms becomes critical. Anomaly detection is crucial in identifying and mitigating potential security threats, network failures, and performance degradation issues.

This state-of-the-art report aims to provide a comprehensive overview of the latest advancements and research trends in anomaly detection techniques for future networks. By examining the current landscape of anomaly detection approaches, we seek to identify the most promising solutions and highlight the key challenges that must be addressed to ensure the security and resilience of next-generation networks.

The objectives of this report are threefold: i) to survey and analyze the existing anomaly detection techniques based on AI and formal methods ii) to identify the strengths, limitations, and potential synergies among these approaches; and iii) to outline future research directions and opportunities in anomaly detection for future networks, like smart home ones.

1.2 Sections Overview

This state-of-the-art report is organized into three sections, each focusing on a specific aspect of anomaly detection in future networks.

Section 2 explores *AI-based* anomaly detection in networks. It introduces key machine learning approaches used for IDSs, outlines the challenges of next-generation networks, and presents recent solutions such as distributed IDS and data reduction. The section also highlights the limitations of current methods and proposes adaptive traffic representation as a promising direction for efficient and scalable IDS.

Section 3 delves into anomaly detection using *runtime verification*. It provides an introduction to runtime verification of distributed systems, and its application in anomaly detection. An overview of existing runtime verification approaches is given: assumptions on the systems, formalizations of specifications and different monitoring architectures are surveyed. The section then discusses the challenges and perspectives for using runtime verification for anomaly detection in future networks.

Section 4 focuses on smart home security and the Matter protocol. It begins with an introduction to smart home security and an overview of proprietary systems. The section then presents a case study of the Matter protocol, examining its network architecture, security and privacy features, and session establishment mechanisms. The state of the art in smart home security and detection technologies is also discussed.

Finally, the report summarizes the main findings and insights from the state-of-the-art survey, highlighting the key research gaps and future research opportunities in anomaly detection for future networks.

2 AI-Based Anomaly Detection

2.1 Introduction

AI has emerged as a powerful tool for anomaly detection in networks, leveraging machine learning (ML) and deep learning (DL) algorithms to identify patterns, anomalies, and potential threats in network traffic. AI-based approaches offer several advantages over traditional methods, such as learning and adapting to changing network environments automatically, detecting unknown anomalies, and handling large-scale, high-dimensional data. However, the emergence of new networks, such as 5G and 6G, adds a layer of complexity to the intrusion detection problem in networks. Current solutions are no longer suitable and must be improved to meet the new challenges.

In the following sections, we 1) present the most commonly used AI solutions for intrusion detection in the literature, 2) describe the new challenges posed by next-generation networks, 3) focus on the problem of very high bandwidth, and explore state-of-the-art solutions that address this issue. 4) We conclude by discussing the need to develop new approaches for describing network traffic.

2.2 State-of-the-art ML Algorithms for Anomaly Detection

In the literature, the most commonly used AI algorithms for intrusion detection are *supervised* approaches [99], which rely on labeled data to train the model. These models learn to classify data points based on the provided labels, making them particularly effective in detecting known anomalies. However, despite their excellent performance, supervised methods are often impractical for real-world scenarios. Their dependence on large amounts of labeled data and the complexity of implementation in dynamic environments make them less realistic for widespread adoption. This limitation partly explains why AI-based IDS solutions remain scarce and underdeveloped in industrial settings [102].

In contrast, *unsupervised* methods do not require labeled data. Instead, they focus on learning the underlying structure of the data and identifying anomalies as deviations from the established patterns. While unsupervised approaches generally yield slightly lower performance than supervised methods, they are more adaptable to real-world scenarios. For instance, a combination of Auto-Encoders and One-class Support Vector Machine (SVM), as demonstrated in the work of [24], have been successfully applied to detect anomalies in network traffic. Given their practical advantages, further exploration of semi-supervised [2]—which rely on a small amount of labeled data combined with a large volume of unlabeled data— and unsupervised methods is highly recommended [93].

AI methods used in intrusion detection can also be categorized into *shallow* and *deep* approaches. Shallow methods, such as decision trees (DTs) or SVMs, rely on handcrafted features and represent classical ML techniques. Deep methods, including deep neural networks and autoencoders, consist of multiple layers that can automatically learn features from raw data.

Current trends highlight a growing interest in deep methods [5] due to their superior performance and ability to uncover complex patterns in data. However, shallow methods remain valuable due to their simplicity, maturity, and interpretability. For example, decision trees and random forests are easier to parameterize and provide transparent explanations for their decisions, as illustrated in the work of [75], which employs decision tree model for intrusion detection. Deep methods, while initially perceived as "black boxes," are increasingly benefiting from advancements in explainability techniques [87]. These developments make deep learning solutions highly effective but also interpretable and understandable, bridging the gap between performance and trustworthiness [3]. Both shallow and deep methods have unique advantages, and their complementary strengths continue to drive innovation in AI-based intrusion detection.

2.3 Challenges for Designing IDS for 5G and Beyond

The advent of 5G and future networks introduces new challenges and requirements for designing effective IDS. While AI-based methods for classification, including intrusion detection, already face significant challenges—such as data imbalance, high false positive rates, and the need for large labeled datasets— these

difficulties are further amplified by the emergence of next-generation networks. These advanced networks bring unique complexities that demand IDS solutions that address novel challenges including the next four ones [31]:

- **C1: Increasing volumes of data**: The massive amount of data generated by future networks poses significant challenges for traditional IDS. Processing and analyzing such vast amounts of traffic in real-time is difficult, often resulting in delayed detections and increased vulnerability to security breaches.
- **C2: High transmission rates**: The ultra-high-speed nature of these networks necessitates IDS that can operate at exceptional speeds to match the data flow. This requirement amplifies the difficulty of achieving reliable real-time intrusion detection.
- **C3: Data structure heterogeneity**: The diverse ecosystem of devices, protocols, and services in next-generation networks generates data in various formats and structures. This heterogeneity complicates feature extraction and anomaly detection, making it harder to achieve accurate results.
- **C4: High mobility**: The mobility of devices within these networks introduces further complexity, as IDS must continuously adapt to dynamic network topologies and rapidly evolving device behaviors.

This project focuses on **C1** and **C2** as they are the most critical. To address them, we provide a detailed state-of-the-art review of intrusion detection methods that are designed to tackle these issues effectively.

2.4 State-of-the-Art of AI-Based IDS for Future Networks

The rapid evolution of future networks, such as 5G and beyond, has spurred significant advancements in AI-based IDS. These systems aim to address the challenges posed by high bandwidth, ultra-low latency, and diverse traffic patterns through innovative techniques, including essentially i) **distributed processing**, and ii) **data reduction methods**.

Distributed processing: By distributing IDS functionality across multiple nodes, this approach enables parallel traffic analysis, reducing latency and improving scalability. For instance, Viegas et al. [111] introduce BigFlow, a distributed intrusion detection system (IDS) designed for high-speed networks. BigFlow leverages stream processing frameworks to perform real-time feature extraction and classification across multiple nodes, ensuring scalability and maintaining high detection accuracy while processing traffic rates up to 10 Gbps. Similarly, Qadeer et al. [92] propose an efficient multicore IDS architecture that distributes packet capturing and processing across multiple cores. Their design incorporates PF_ring, an efficient packet capturing library, and achieves load balancing using IP hash. By employing 16 cores, their system can process network traffic at rates exceeding 1 Gbps. These distributed architectures significantly reduce computational overhead, minimize packet loss, and enable real-time threat detection in high-speed network environments.

Data reduction techniques: Advanced data reduction methods, such as feature selection and dimensionality reduction, are also employed to reduce the volume of data that needs to be processed by the IDS. These methods aim to identify the most relevant features for anomaly detection, discarding redundant or irrelevant information. By reducing the data dimensionality, the IDS can process data more efficiently, enabling real-time detection in high-speed networks. This area of research has been extensively explored in the literature. For example, Nabi and Zhou [86] compare the impact of Principal Component Analysis (PCA) and Random Projection (RP) on the classification accuracy of IA-based IDSs. Their results demonstrate that Random Projection outperforms PCA for most classifiers while requiring significantly less computation time. In another study, the authors of [114] propose using a deep learning approach, specifically the Stacked Sparse Autoencoder (SSAE), for feature extraction in intrusion detection systems. The authors introduce the original classification features into the SSAE to automatically learn high-level, low-dimensional sparse feature representations of intrusive behavior information. These learned features are then used to train various basic classifiers.

Another promising direction for addressing the challenges of real-time detection (C1) and large-scale data processing (C2) in AI-based IDS is **the way network traffic is represented**. The representation of

network data directly impacts the performance of IDS, particularly in terms of detection speed and scalability. A well-chosen representation can significantly reduce processing overhead while preserving the quality of information needed for accurate classification.

Two main approaches to traffic representation are commonly used:

- **Packet-based:** This method analyzes individual network packets, capturing fine-grained information and enabling precise anomaly detection. However, due to the high volume and granularity of data, packet-based analysis is computationally intensive and may struggle to scale in high-speed environments such as 5G networks.
- **Flow-based:** This method aggregates packets into flows, offering a summarized view of traffic that reduces the computational load. While more scalable and faster to process, flow-based approaches may miss low-level anomalies that are only detectable at the packet level.

To balance the trade-offs between granularity and efficiency, recent research has proposed **hybrid traffic representation** strategies that combine the strengths of both approaches. For instance, Kim and Pak [66] propose a two-stage IDS architecture: initial real-time detection is performed at the **flow level** using a lightweight, fast decision algorithm. When necessary, the system then triggers a **packet-level** analysis for suspicious flows, applying a more thorough but slower method. This design prioritizes speed without compromising accuracy, reducing memory usage by up to 30% compared to traditional approaches.

Similarly, Seo and Pak [103] introduce a two-level classification scheme in their hybrid intrusion prevention system. At Level 1, a high-speed classifier handles traffic with fast decisions based on **flow-level** features. If the confidence level is low, the traffic is escalated to Level 2, where a more precise, **packet-based** classifier ensures detailed inspection. This tiered approach allows the system to maintain real-time responsiveness while significantly improving detection reliability.

2.5 Adaptive Traffic Representation for efficient IDS in Future Networks

The representation of network traffic data remains a fundamental challenge for intrusion detection in future networks. Packet-based and flow-based approaches have strengths and limitations, and hybrid methods offer a promising pathway. However, most current hybrid approaches follow fixed decision rules that do not adapt to changing network conditions. This lack of flexibility reduces their effectiveness in dynamic and high-speed environments, where the optimal level of data granularity may vary. For instance, packet-level inspection may be suitable during low traffic periods due to its precision, while flow-level aggregation is more appropriate under heavy loads for better scalability.

A promising avenue for future work involves the development of *adaptive traffic representation strategies.* These would dynamically select the appropriate level of detail—packet or flow—based on real-time traffic characteristics, system load, and detection requirements. For example, an adaptive IDS might use detailed packet analysis when anomalies are suspected, while relying on flow-level summaries during normal conditions to reduce computational cost.

Key challenges include designing algorithms capable of dynamically switching between packet-level and flow-level representations, ensuring that adaptive techniques maintain high detection accuracy without introducing excessive computational overhead, and integrating these methods into existing AI-based IDS frameworks for seamless operation in high-speed networks. Addressing these challenges is essential to meet the scalability and efficiency requirements of next-generation IDS and ensure robust security for future networks.

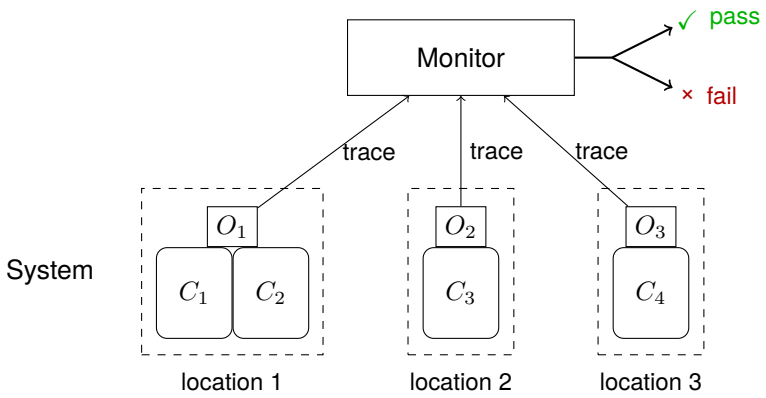


Figure 1: Example of a runtime verification set-up for a distributed system with four components at three locations.

3 Anomaly Detection using Runtime Verification

Future networks, such as 5G networks, are distributed systems that are often large and open, making them susceptible to attacks and failures. Our goal is to define new approaches for detecting attacks at operation time. Section 2 explored solutions based on AI to identify intrusions. In this section, we explore another family of approaches, called *runtime verification*, which is based on formal methods and that we will also use for intrusion detection. Runtime verification consists of confronting system executions to a formal reference, i.e. a specification, to identify specification violations in these executions. We intend to define system specifications so that their violations can be considered hints of intrusion occurrences.

3.1 Introduction to Runtime Verification

Runtime verification focuses on analyzing system executions at runtime. More precisely, executions are checked against a *specification*, which formalizes either a desired property of the system or a model of the expected behavior of the system. This process occurs in two steps which can be intertwined: (1) System executions are *observed* and pieces of information, relevant to the analysis, are recorded into sequences of events called *traces*; (2) One or several components, called *monitors*, are in charge of checking that the traces comply with the specification. If they do not, the monitor sends an error message.

In the context of this project, we pay attention to distributed systems, that is, systems composed of several sub-systems, or components, deployed on different locations and which collaborate by means of message-passing. The reason of this focus is that we are interested in adapting runtime verification to detect potential network intrusions, whose effects are observable as communication protocol violations. In this context, we are interested in techniques where: (1) observations can occur simultaneously at different locations and, (2) the monitoring phase can confront these different observations to identify protocol violations (possibly caused by malicious actions).

Figure 1 illustrates a runtime verification scenario where the system under observation is made up of four components C_1, C_2, C_3 and C_4. These are distributed over three locations, with components C_1 and C_2 at the same location. Observation interfaces O_1, O_2, O_3 are placed at each location. They record the traces at their location and send these to a monitor for analysis. The monitor checks the received traces against the specification and answers pass or fail.

3.2 State-of-the-Art

Runtime verification has its roots in different research fields of formal methods such as model-checking, model-based testing, and process algebras. We now provide an overview of the literature on runtime ver-

ification, with a focus on the aspects most relevant to our objectives. Since we target systems leveraging future networks, we are specifically interested in runtime verification techniques for distributed systems. Additionally, given that our primary goal is to detect intrusions, we will place particular emphasis on techniques that enable the analysis of communication flows.

We start by reviewing the different assumptions made about the distributed systems (in our case, the networks) we work with. We then examine the different formalizations for writing specifications (in our case the expected network behaviour), and finally explore the existing monitoring solutions.

3.2.1 Characteristics of Distributed Systems

Distributed systems are assumed to have some form of *message-passing* between components. That is, components can send messages which can be received by some or all other components. More or less information may be known about the structure of the systems under observation. In some cases, the only information are the events that can be observed at runtime (for example using a Wireshark to capture packets at certain locations of the system). Sometimes there is information on the communication architectures between components; for example, whether they are lossy, i.e. whether messages may be lost. In [53, 84], the authors assume lossless FIFO channels, which implies that the system components receive messages in the order in which they are sent, and no messages are lost. Systems components may be known in more detail: in some papers they are defined using processes, as in the field of process algebras (e.g. in [23,25]). These are small program descriptions with internal events, variable assignations and message passing to other processes.

A *global clock* may be assumed, as in [33, 44, 46]. This allows the events of the different components to be totally ordered into a *global trace*. A global clock may be reasonable in some settings, e.g. when the components execute on several cores of the same CPU, and it allows for many precise runtime verification techniques. However, for large distributed systems with components in different locations, the assumption of a global clock is often not realistic. One must then make use of the local clocks which order events on co-located components.

Several solutions for dealing with separate clocks appear in the literature. Vector clocks [50, 71, 82] are used to synchronize local clocks: the distributed components each keep a vector with their knowledge of other components' local clocks. Every time a component sends a message, it appends its vector, and every time a component receives a message, it updates its vector with the information received in the message. This allows the total ordering of all system events, but comes with a high communication overhead. It is used for example in [84, 100]. Another approach, seen in papers [52, 53], is to assume that the distributed system has a clock-synchronization algorithm which bounds the skew among the local clocks of the different components; this setting is called *partial synchrony*. One may also work directly with *multi-traces*, that is the collection of local traces, without trying to order them, and adapt the runtime verification approach accordingly (e.g. [76]).

3.2.2 Specifications

We want to check a real system, via the (multi-)traces we observe, against a model of the system that specifies what should happen. For example, consider a network in which the network nodes interact via a message-passing protocol. This protocol constitutes a model of the expected behavior for communication in the network. The formalization of an expected behaviour is called a *specification*.

Several formal languages for writing specifications exist. A popular language for writing specifications is linear temporal logic (LTL) [91]. LTL is evaluated over infinite traces: given an LTL formula, infinite traces either satisfy the formula or do not. However, in runtime verification, finite traces are observed and evaluated. In general the system under observation continues running, therefore these finite traces may be seen as prefixes of infinite traces. To accommodate this, a three-valued modification to the semantics of LTL was introduced, called LTL_3 [21]. Given a finite trace and a formula, the trace evaluates to $\top$ if all infinite continuations of the trace satisfy the formula, $\bot$ if none of them do, and $?$ otherwise; $\top, \bot, ?$ are called *verdicts*. Runtime verification papers using LTL or LTL_3 with verdicts include [33, 44, 53, 84, 97]. Verdicts can also be

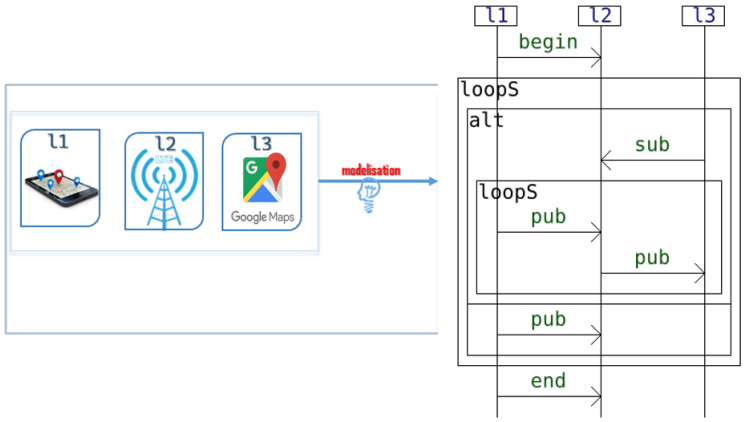


Figure 2: A specification formalized as an interaction.

used for specifications that are not LTL but regular languages, as in [46]. Other variations on LTL appear in runtime verification papers, like PT-DTL [101] which is used for specifications across different locations of a distributed system, and which talk about other components as well as about past events.

Another line of work uses the multi-party session types (MPST) framework [59] to write and enforce specifications for distributed systems. Paper [25] writes specifications as global types, which are then projected into local types checked by local monitors attached to a single component.

Interaction models are a class of specification formalisms which are designed to be easy to write and understand from an engineering perspective. They describe the communication flow between different components of a distributed system. They include UML [26], message sequence charts (MSC) [83] and interactions [78]. UML is extensively used as a modeling language in applications, but its semantics are not defined in a way suitable for formal verification methods. [9, 69] use MSCs as specifications. Interactions are used for runtime verification in a recent line of work [76, 77, 79]. Other specification formalisms used in runtime verification papers are Petri nets, a classic model for concurrent systems used here to model desired behaviour [64], and trace expressions [11–13].

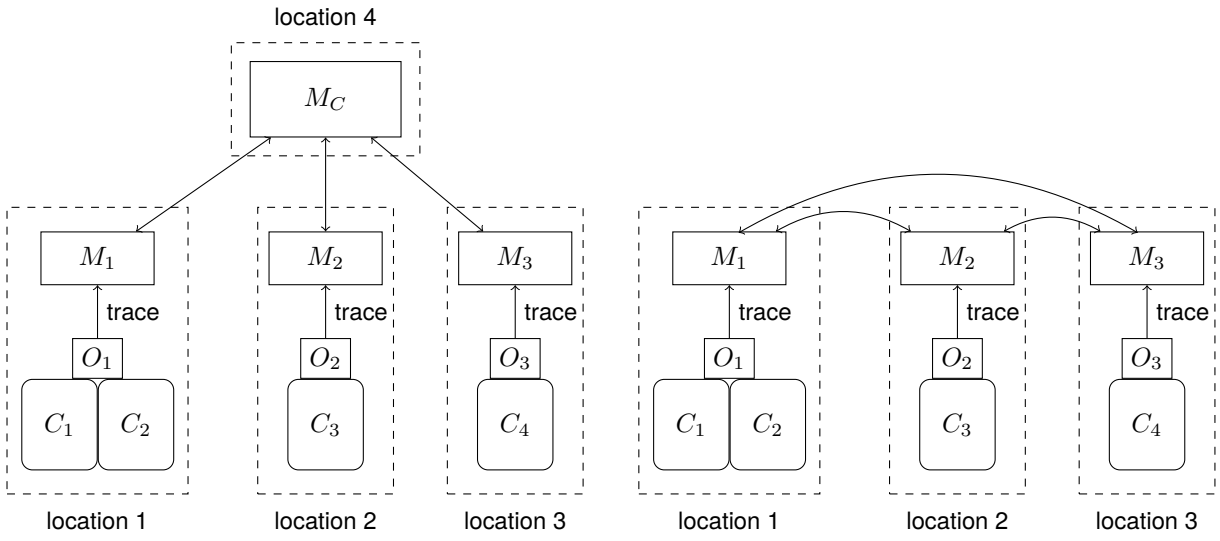
Figure 2 shows a specification formalized as an interaction. This interaction describes a toy MQTT-style publish-subscribe protocol between a phone, a base station and a server for Google Maps. These three components are represented by three *lifelines* l_1, l_2, l_3 that interact via message-passing. The modeled behavior is a session that starts with message `begin` sent from l_1 to l_2, then a loop with two alternative behaviors, and finally a message `end` from l_1 to l_2.

3.2.3 Monitoring Solutions

Runtime verification may be *offline* or *online*. In offline runtime verification, the system's execution traces are logged at runtime and checked against the specification at a later time. This implies that full traces are recorded, and that the complexity of checking the traces will not interfere with the system. Some examples of papers that analyze traces offline are [76, 88, 97].

In online runtime verification, execution traces are checked while the system is running. The runtime verification machinery (observation, recording, monitoring) must be considered as part of the system, which entails considerations absent from offline runtime verification. For instance, one must take into account the difference between the speed at which traces are recorded and the speed of analysis by the monitor. Traces are recorded into a structure, for example a buffer, which is necessarily finite. Either the monitoring speed must be faster than the recording speed, or the monitor must be designed to deal with loss of data.

Different monitoring architectures are possible for online runtime verification. As shown in Figure 1, there may be one *centralized* monitor which receives the traces to be checked against the specification (see e.g. [20, 21, 100]). Alternately, there may be local monitors that perform checks locally. This strategy is called



(a) Orchestrated monitoring.

(b) Choreographed monitoring.

Figure 3: Distributed monitoring architectures.

decentralized monitoring in the presence of a global clock, and *distributed* monitoring in the absence of a global clock (though the terms are sometimes used interchangeably).

In *orchestrated* monitoring, the local monitors communicate with a central monitor. In *choreographed* monitoring there is no central actor and the monitors exchange messages with each other. Figure 3a shows an orchestrated monitoring scenario, while Figure 3b shows a choreographed monitoring scenario. Monitors may communicate with each other through their own channels [46], or they may “piggyback" the existing message framework: when process P sends a message to another process, the monitor in the same location as process P adds information into the message being sent, for example a vector clock or the result of its local computation (e.g. [84, 101]).

When a trace does not satisfy the specification, the monitors may send an error message or interrupt the computation. The monitors may also *enforce* the correct behaviour at runtime by not letting the system perform bad actions, as is the case in [25]. A classification of existing runtime verification tools (up until 2021) can be found in [47].

3.3 Challenges and Perspectives

5G networks and future networks are an ideal application domain for runtime verification techniques, which are developed for distributed systems. We want to be able to detect as many deviations as possible from the expected flow of communication. We propose to write our specifications as interactions, as these are well suited for describing communication protocols. We will develop online runtime verification techniques, since these sorts of networks have ongoing communication flows and we want to raise errors as soon as possible. To deal with the speed of communication, we propose to look at distributed monitoring architectures: monitors placed at nodes of the network can perform efficient local checks, and additional communication with other monitors allows the detection of global errors. Additional challenges that we will keep in mind when designing our solutions are (1) the potential loss of data due to the speed of communication versus the speed of verification, and (2) the mobility of devices in dynamic networks, implying the need to design solutions in which sub-components may join or leave the system.

4 Smart Home Future Network

4.1 Introduction

A *smart home* network, also known as *home automation* or *domotic* network, monitors and controls a home using connected smart devices like smartphones, hubs, cameras, water sensors, and doorbells. It typically includes point-to-point links and an Internet connection. It employs various communication technologies, such as Wi-Fi, Bluetooth Low Energy, and Thread, as well as cloud backends from vendors like Amazon, Google, and Apple.

As shown in Figure 4, a smart home network comprises local devices, one or more gateways, and a cloud backend. The local device communicates in point-to-point or mesh (multi-hop) networks. They can use wireless access points or wired networks. The gateways translate among different protocols and improve interoperability. For example, a Thread border router connects devices in a low-power mesh network with those in the local area network. Different cloud backends must communicate to enable seamless device control. For instance, a Google smart controller sends control commands to the Google cloud, which communicates to the smart-controlled device cloud and relays the command.

Early development of smart home networks and related pros and cons can be traced back to the beginning of the 2000s. MIT's House_n [62] was centered around context-aware sensing and presentation of information. The adaptive home presented a house that programs itself (other than a programmable house) [85]. eHome [68] presented a case study focusing on the usability of three smart home user interfaces (a PC, a smartphone, and a media terminal). In [113], the authors conducted a case study of home automation usage for religious purposes on 20 American Orthodox Jewish families. The Georgia Tech aware home [65] was a multi-disciplinary attempt at designing a smart home emphasizing families and aging people. These early attempts did not find broad adoption mainly because of four reasons: 1) high monetary and time cost of ownership, inflexible devices, poor manageability, and difficulty in achieving security [28, 95]. We focus on the last challenge, i.e., *smart home security*, and also *smart home privacy*.

Smart home devices' abundance, heterogeneity, and connectivity result in a *massive attack surface* with critical associated security and privacy risks. For example, an attacker can try to remotely tamper with a smart home device exposed to the internet or reachable via the cloud. They can also try to connect to the local area network and conduct passive and active attacks locally. Finally, the attacker can target the system with proximity-based attacks using wireless technologies or physical attacks. Smart home networks have historically been *proprietary*, but they are moving to *open* architecture. Next, we describe both approaches and the related state of the art.

4.2 Propietary Systems

There are four major smart home players: Google, Amazon, Samsung, and Apple. Each provides a proprietary and historically difficult-to-interoperate smart home system. The situation has improved with the introduction of Matter, a standard for interoperable smart home networks, which is discussed in Section 4.3. Next, we describe the four major smart home players and survey the state of the art of proprietary smart home security and privacy.

Google Home. Google's smart home ecosystem is centered around the Nest hubs and smart speakers [55]. These devices can manage and control smart home devices using the Google Assistant voice-enabled system. Google also offers a Home application for Android and iOS that collaborates with the Google Nest devices—a smart home device compatible with Google Home, as a dedicated logo.

Amazon Alexa. Amazon's smart home ecosystem relies on the Echo and Tap smart speakers. These devices act as hubs, similar to Google's, and are compatible with Alexa, Amazon's virtual assistant. Many smart home devices are compatible with Amazon's smart home, which are sold in the Amazon marketplace [10].

Samsung SmartThings. Samsung SmartThings [98] has three main components: a hub, a backend, and a SmartThings mobile application for Android or iOS. The hub supports ZWave, ZigBee, and Wi-Fi and can interact with the devices in proximity. The smartphone companion app can communicate with the hub,

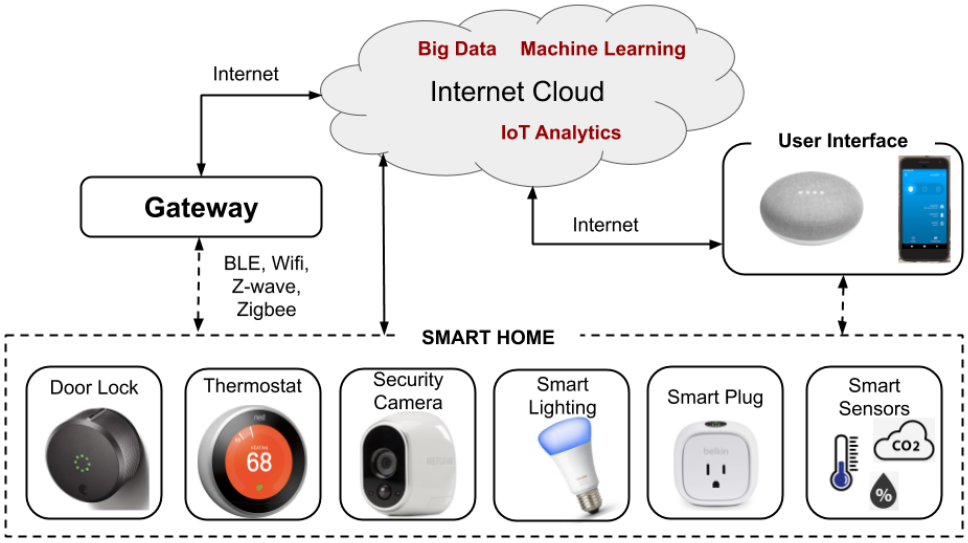


Figure 4: Smart home network architecture comprises local devices, gateways, and cloud backends.

manage the smart devices, and install SmartApps on the devices from a dedicated store. The hub and SmarThings communicate using a proprietary application layer protocol over TLS with the backend.

Apple Home. Apple Home [16] is a smart home platform developed by Apple. It is based on the HomeKit framework and related APIs [17]. It is compatible with Siri, Apple's voice assistant, and the iOS, macOS, and iPadOS Apple OSes. It enables developers of mobile applications and smart home devices to take advantage of Apple's protocols (HAP) and services.

4.2.1 Smart Home State-of-the-Art

Before the Matter standard, discussed in Section 4.3, several works analyzed the security and privacy of proprietary and competing smart home systems. Dennis [41] presented a framework for evaluating security risks in the modern home, including new attacks and known ones with novel consequences. Ur [107] presented a study on smart home access control focusing on Philips Hue lighting and Kwikset door lock, demonstrating the challenges of shared control with proprietary systems.

Fernandes in [48] discussed a security evaluation of Samsung SmartThings. They analyzed 499 SmartApps and 132 device handlers and uncovered two design flaws that led to overpermissioning attacks, including leaking a door lock. Then, Fernandes proposed FlowFence [49], a system based on a finer-grained permission model, as a defense against the prior attack. In [57], the authors concluded from a case study involving 425 participants that a smart home access control policy should be based on device capabilities other than devices.

Wang in [112] discussed ProvThings, a tool to instrument IoT home automation apps to add centralized audit logging capabilities, and tested its efficacy against 26 IoT attacks on the SmartThings platform. Alrawi in [8] presented a methodology based on scorecards to analyze and systematize the security of a home automation system and evaluated it on 45 IoT devices. In [116], the authors compared five smart home platforms and modeled their cloud, IoT, and mobile app components using state machines.

Kumar [70] described a large-scale case study involving 83M devices and 16M households, showing that home automation is widespread across continents and involves heterogeneous devices. Moreover, they show that IoT security weaknesses vary across manufacturers and within a manufacturer based on the device's geographical location. Similarly, the IoT Inspector work [61] shows that by collecting an extensive and labeled dataset of smart home communication, one can infer bad security and privacy issues across vendors, including the use of weak Transport Layer Security (TLS) versions or data exfiltration to advertising

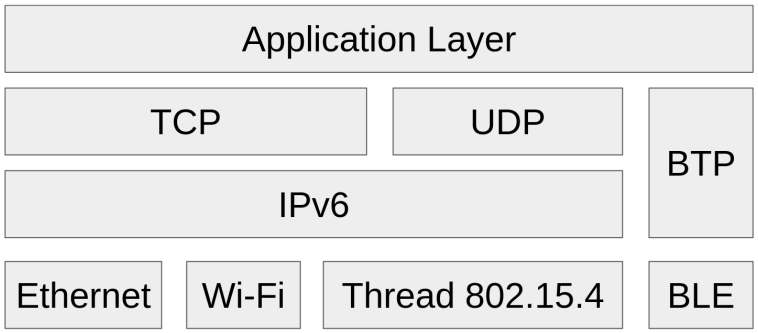


Figure 5: Matter stack centered around IPv6. Bluetooth Low Energy (BLE) and Bluetooth Transport Protocol (BTP) are used only for discovery and PASE.

services. In [4], the authors show a multi-stage attack based on machine learning methods to infer smart home device data even if it is encrypted.

Proprietary smart home management systems were analyzed in [63] and shown to be uncoordinated and vulnerable to access control attacks. In [115], the authors systematize research related to privacy-preserving smart hub devices from 10 industrial proposals and 37 research papers. Privacy techniques include packet obfuscation and local data processing, minimization, and obfuscation. Mandalari in [80] evaluated eight popular IoT security services, mostly provided via commercial routers, to protect smart homes and found that they provide limited security and privacy protection even against basic attacks and might introduce extra privacy risks.

Proprietary wireless protocols for home automation like Z-Wave [51, 110] and ZigBee [32, 96], have been found vulnerable as well. Researchers presented, among others, battery depletion and impersonation attacks.

4.3 Matter Smart Home Standard

Matter is a standard technology for interoperable, secure, private, and usable *smart homes* [39]. It is centered around IPv6 and allows the management of heterogeneous devices, including cameras, vacuum robots, access points, energy and light management, smoke and carbon monoxide detectors, presence sensing, and closure sensors. It does not require an Internet connection to operate. Matter was formerly known as Project Connected Home IP (CHIP).

Matter is specified in a set of open documents maintained by the *Connectivty Standards Alliance (CSA)*, a consortium with more than 200 companies, including Apple, Google, Amazon, Comcast, IKEA, Huawei, and Schneider [39]. The Matter v1.0 standard was introduced in October 2022 [35] and is updated biannually. The paper refers to Matter *v1.4*, the latest standard version. The CSA also provides an official and open-source Matter Software Development Kit (SDK) [37] and related documentation [36], including a reference Matter implementation and examples.

Matter employs two secure session establishment protocols called *Passcode Based Session Establishment (PASE)* and *Certificate Based Session Establishment (CASE)*. PASE is a Password-Authenticated Key Exchange (PAKE) protocol used to generate an authenticated session key to protect the Matter onboarding process (i.e., Matter commissioning). The protocol protects against online and offline dictionary attacks while allowing low-entropy static or dynamic passcodes. CASE is a SIGn-and-MAc (SIGMA) based protocol generating authenticated session keys using X.509 certificates. The Matter commissioner distributes the certificates and acts as the Certificate Authority (CA).

4.3.1 Matter Network

Matter aims to standardize universal smart home technology based on IPv6. As shown in Figure 5, Matter supports wired and wireless *link layers* for device interoperability, including Ethernet, Wi-Fi, Thread, and BLE. Being centered around the IPv6 network layer, it is compatible with all protocols running over IPv6, including the TCP and UDP transport layer protocols. It defines a Matter *application layer* protocol based on a standard data model. Moreover, it provides two custom TCP-like transport protocols: BTP to talk Matter over BLE (MATTERoBLE) and Message Reliability Protocol (MRP) to add reliability to UDP.

A Matter network is called *fabric* and is identified by a unique 64-bit Fabric ID. A fabric is not centrally owned, but it is shared among devices. Each device connects to the fabric using one network interface, like `wlan0`. A unique 64-bit Node ID identifies a device or a group of them within a fabric. A fabric can be established without an Internet connection and behind a firewall (i.e., no globally routable IPv6 infrastructure is needed). It can cover one or multiple IPv6 subnets. For example, a fabric can include a Wi-Fi network with Internet access bridged to a Thread network for low-power and long-range mesh communication. The two networks are bridged by a *Matter Border Router*.

Matter defines specific *roles* to establish and manage a fabric [38, Sec. 1.3]. A *Node* is a Matter entity with one or more addressable *Endpoint* and can have multiple roles. A *Device* includes one or more Nodes. A Node can be a *Commissioner*, a role able to commission (i.e., let other devices join) the fabric. It can be *Commissionable*, a role to join the fabric. The latter becomes a *Commissionee* while it is being commissioned. A commissioned device acts as *Controller* if configured to control other Nodes or *Controlee* if configured to be controlled. The mapping between these terms and HomeKit, Weave, Thread, and Zigbee is provided in [38, Sec. 1.4].

A Node can join multiple fabrics, each with a unique set of administrators, permissions, and settings. For example, a smart home device, like a thermostat, could be part of a home and service provider's fabric. The homeowner sets up and controls the former, allowing household members to control the temperature settings. The latter is managed by a utility company for remote diagnostics or energy-saving programs.

4.3.2 Security and Privacy

Matter *security* relies on five properties [34]: i) *Comprehensive protection* provides self-contained device authentication and attestation, encrypted communication, and secure firmware updates. ii) *Strong mechanism*, such as AES-CCM, SHA256, and ECC over secp256r1, ensure that Matter takes advantage of standard and battle-tested cryptographic ciphers and protocols. iii) *Easy to use* security aspects enable vendors and users to take advantage of standard and reference implementations without creating potentially vulnerable solutions. iv) *Resilient approaches* are used to protect, detect, and recover from a security or privacy incident. v) *Crypto agility* enables the update of the Matter cryptographic primitives and protocol to address future threats without breaking the specification.

Matter *privacy* is guaranteed by the four properties [34]: i) *Confidentiality* of data in transit, ii) *Device authentication* using x.509 certificates signed by a trusted CA, iii) *Data minimization* to minimize the amount of information exchanged during a session iv) *Privacy-preserving mechanisms* in the standard, including unique random node IDs, non-trackable IP addresses, and encrypted session metadata.

The Matter standard discusses threats and countermeasures [38, Sec. 13.7]. Each threat has an ID, description, threat agent (attacker model), threat evaluation (impact and severity), and countermeasure. The threat list includes impersonation and machine-in-the-middle threats. Next, we describe Matter's state of the art.

4.3.3 Matter State of the Art

There are a few research papers about Matter. Next, we present them based on their category.

Survey. Authors in [22] provide a concise description of the Matter standard, survey research work by academia and industry, and offer insights on addressing current limitations.

Discovery. In [18] the authors present a detailed analysis of the Multicast DNS (mDNS) and DNS-Based Service Discovery (DNS-SD) specifications and discuss their attack surface including eavesdropping, impersonation, and Machine-in-the-Middle (MitM) threats. Attacks and defenses on proprietary Apple services using mDNS and DNS-SD, including AirDrop, were presented in [19, 43, 106]. Other prior work touched on similar issues [54, 90], including getting mDNS responses from outside the local network using a unicast query [30].

Attacks. Several research works presented theoretical and practical attacks on Matter. In [104], the authors highlight a security issue in Matter's commissioning process, enabling unauthorized access via an unenrolled channel. They tested 15 devices, underscoring IoT risks, and recommend enforcing a single active device channel to enhance IoT security. In [72], the authors highlight the Matter standard's device pairing and delegation process vulnerabilities. They introduce the Hidden Eavesdropping Attack, in which unauthorized hubs exploit flaws to eavesdrop on IoT devices, exposing sensitive data and compromising user privacy. One paper studied low-level jamming attacks on Thread, focusing on OpenThread [56] and its jamming detection mechanism [67].

Standard security. Other research analyzed the security of the Matter standard. In [40, 73], the authors evaluate the security of the Matter standard through threat modeling and vulnerability analysis, highlighting potential weaknesses, vendor-specific risks, and deviations from security promises while providing recommendations and symbolic verification models to enhance Matter's security framework. In [105], the authors examine Matter's design, where controllers are not required to verify their trustworthiness to devices. Through experimentation, the authors highlight scenarios where malicious controllers could harm the Matter system.

Communication. Significant research efforts have been allocated to Matter's Wi-Fi, Thread, and Bluetooth communication technologies. What follows is a sample from the many research works in these fields. Wi-Fi security mechanism, including WEP, WPA, WPA2, and WPA3 have been found vulnerable to impactful attacks [27, 108, 109]. Regarding Thread, its side channel resistance against differential EM analysis was assessed in [42]. Thread's commissioning protocol, called J-PAKE, was formally modeled and analyzed [1]. Moreover, in [6], the authors study battery depletion and online password-guessing attacks on a Thread network. MUDThread [60] shows how to integrate the Manufacturer Usage Description (MUD) IoT standard into a network of constrained Thread devices. BLE has been found vulnerable to several therats, including the KNOB [14] and BLUR [15] attacks.

Testbed. In [81], the authors examine the Matter protocol's role in enhancing smart home interoperability, introducing a network utility device to analyze IoT network traffic, and providing insights from an academic testbed setup.

Fuzzing. In [74], the authors introduce mGPTFuzz, utilizing large language models to automate test input generation based on Matter's extensive specification. Evaluated on 23 devices, mGPTFuzz discovered 147 new bugs, including three CVEs, outperforming existing IoT fuzzers.

4.3.4 Smart Home Intrusion Detection

Several works propose generic intrusion and anomaly detection/prevention techniques for the smart home. Ramapatrun [94] explores machine learning techniques to identify anomalies in a smart home. The presented Hidden Markov Model (HMM) achieves an accuracy of 97% when trained with sensor data from the network. Reinforcement learning was also considered in smart home intrusion detection, and MAGPIE was proposed as an IDS capable of dynamically adjusting its detection logic based [58]. Power consumption was also used as a metric for anomaly detection attacks [89]. Hybrid approaches for anomaly detection were also discussed. In [7], the authors evaluate the CSE-CIC-IDS2018 public dataset with different Machine Learning (ML) classification algorithms, including random forest, xgboost, and decision trees. For a survey on the field, please refer to [45, 115].

However, *there is no standardized intrusion detection technology for Matter*. The standard relies on its security and privacy mechanisms and protocols, such as CASE and PASE, to defend the network from threats such as impersonation and MitM threats.

5 Conclusion and Future Directions

AI-based anomaly detection has made significant strides in recent years, providing powerful tools for securing next-generation networks (Cf. Section 2) . However, the emergence of 5G and beyond has exposed key limitations, particularly in how network traffic is represented. Although packet-based, flow-based, and hybrid methods each offer strengths, they fall short in fully capturing the complexity of modern traffic. Future research should focus on adaptive and dynamic traffic representation to meet the demands of high-speed, heterogeneous, and evolving network environments, enabling more scalable and effective intrusion detection systems.

Runtime verification techniques, as described in Section 3, are well-suited for anomaly detection in future networks. They analyze execution traces at runtime to detect violations of a specification, offering exact results with formal guarantees. Among the many existing approaches, we propose to focus on distributed runtime verification architectures: local monitors analyze local executions efficiently, and communicate with the other monitors to catch global errors. We will formalize our network specifications as interactions. These are designed for expressing communication flows, and are easy to use and understand. The challenge resides in bringing these solution elements together and designing an efficient procedure which can deal with a high speed of communication, eventual losses of data and nodes leaving and entering the network.

In Section 4 we introduce future networks for smart homes. We survey the state of the art about (legacy) proprietary systems and standard ones based on Matter. As stated in Section 4.3, there is no standard intrusion detection technology for Matter. This is problematic, as the standard is used on billions of devices. We aim to address this gap by proposing a novel IDS for Matter and testing it in a smart home testbed with actual devices. We envision an IDS taking advantage of the AI and runtime verification solutions discussed in Sections 2 and 3. Moreover, we will include traffic analysis techniques capable of (probabilistically) detecting anomalities as we did in prior work on IoT electric scooters [29].

ML-based intrusion detection offers considerable adaptability and can identify anomalous behaviors, including those not explicitly defined in advance (Section 2). Nonetheless, supervised approaches often require large, labeled training datasets, which can be challenging to get. In addition, supervised and unsupervised techniques may produce false positives, potentially impacting the reliability of alerts. In contrast, the runtime verification techniques presented in Section 3 monitor system behavior against formally specified properties. These methods do not rely on training data, and each alert is directly tied to a concrete violation of the specified behavior, resulting in greater precision and fewer false positives. However, their effectiveness is inherently limited to the scope and quality of the specifications, which domain experts must generally craft.

Investigating the complementarity of these two intrusion detection paradigms represents a promising avenue for enhancing intrusion detection systems' overall effectiveness and robustness. Moreover, testing these two techniques in a real-world use case, such as a Matter smart home network (Section 4), would be highly beneficial.

Acknowledgement

This work has been partially supported by the French National Research Agency under the France 2030 label (NF-HiSec ANR-22-PEFT-0009). The views reflected herein do not necessarily reflect the opinion of the French government.

References

[1] Michel Abdalla, Fabrice Benhamouda, and Philip MacKenzie. Security of the J-PAKE password-authenticated key exchange protocol. In *2015 IEEE Symposium on Security and Privacy*, pages 571–587. IEEE, 2015.

[2] Mohamed Abdel-Basset, Hossam Hawash, Ripon K Chakrabortty, and Michael J Ryan. Semi-supervised spatiotemporal deep learning for intrusions detection in iot networks. *IEEE Internet of Things Journal*, 8(15):12251–12265, 2021.

[3] Zakaria Abou El Houda, Bouziane Brik, and Lyes Khoukhi. "why should i trust your ids?": An explainable deep learning framework for intrusion detection systems in internet of things networks. *IEEE Open Journal of the Communications Society*, 3:1164–1176, 2022.

[4] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. Peek-a-boo: I see your smart home activities, even encrypted! In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 207–218, 2020.

[5] Zeeshan Ahmad, Adnan Shahid Khan, Cheah Wai Shiang, Johari Abdullah, and Farhan Ahmad. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1):e4150, 2021.

[6] Dimitrios-Georgios Akestoridis, Vyas Sekar, and Patrick Tague. On the security of thread networks: Experimentation with OpenThread-enabled devices. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 233–244, 2022.

[7] Faisal Alghayadh and Debatosh Debnath. A hybrid intrusion detection system for smart home security. In *2020 IEEE International Conference on Electro Information Technology (EIT)*, pages 319–323. IEEE, 2020.

[8] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. Sok: Security Evaluation of Home-Based IoT Deployments. In *2019 IEEE symposium on security and privacy (sp)*, pages 1362–1380. IEEE, 2019.

[9] Rajeev Alur, Kousha Etessami, and Mihalis Yannakakis. Realizability and verification of MSC graphs. *Theor. Comput. Sci.*, 331(1):97–114, 2005.

[10] Amazon. Amazon Smart Home. `https://www.amazon.com/smart-home/s?k=smart+home`, 2025.

[11] Davide Ancona, Sophia Drossopoulou, and Viviana Mascardi. Automatic generation of self-monitoring mass from multiparty global session types in jason. In Matteo Baldoni, Louise A. Dennis, Viviana Mascardi, and Wamberto Weber Vasconcelos, editors, *Declarative Agent Languages and Technologies X - 10th International Workshop, DALT 2012, Valencia, Spain, June 4, 2012, Revised Selected Papers*, volume 7784 of *Lecture Notes in Computer Science*, pages 76–95. Springer, 2012.

[12] Davide Ancona, Angelo Ferrando, and Viviana Mascardi. Comparing trace expressions and linear temporal logic for runtime verification. In Erika Ábrahám, Marcello M. Bonsangue, and Einar Broch Johnsen, editors, *Theory and Practice of Formal Methods - Essays Dedicated to Frank de Boer on the Occasion of His 60th Birthday*, volume 9660 of *Lecture Notes in Computer Science*, pages 47–64. Springer, 2016.

[13] Davide Ancona, Angelo Ferrando, and Viviana Mascardi. Mind the gap! runtime verification of partially observable mass with probabilistic trace expressions. In Dorothea Baumeister and Jörg Rothe, editors, *Multi-Agent Systems - 19th European Conference, EUMAS 2022, Düsseldorf, Germany, September 14-16, 2022, Proceedings*, volume 13442 of *Lecture Notes in Computer Science*, pages 22–40. Springer, 2022.

[14] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. Key negotiation downgrade attacks on Bluetooth and Bluetooth Low Energy. *ACM Transactions on Privacy and Security (TOPS)*, 23(3):1–28, 2020.

[15] Daniele Antonioli, Nils Ole Tippenhauer, Kasper Rasmussen, and Mathias Payer. BLURtooth: Exploiting cross-transport key derivation in Bluetooth Classic and Bluetooth Low Energy. In *Proceedings of the 2022 ACM on Asia conference on computer and communications security*, pages 196–207, 2022.

[16] Apple. Apple Home App: The Foundation of a Smarter Home. `https://www.apple.com/home-app/`, 2025.

[17] Apple. Apple HomeKit Framework. `https://developer.apple.com/documentation/homekit`, 2025.

[18] Antonios Atlasis. An Attack-in-Depth Analysis of Multicast DNS and DNS Service Discovery. `https://sfc6326dbff511243.jimcontent.com/`, 2017.

[19] Xiaolong Bai, Luyi Xing, Nan Zhang, XiaoFeng Wang, Xiaojing Liao, Tongxin Li, and Shi-Min Hu. Staying secure and unprepared: Understanding and mitigating the security risks of Apple zeroconf. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 655–674. IEEE, 2016.

[20] Fabio Barbon, Paolo Traverso, Marco Pistore, and Michele Trainotti. Run-time monitoring of instances and classes of web service compositions. In *2006 IEEE International Conference on Web Services (ICWS 2006), 18-22 September 2006, Chicago, Illinois, USA*, pages 63–71. IEEE Computer Society, 2006.

[21] Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for LTL and TLTL. *ACM Trans. Softw. Eng. Methodol.*, 20(4):14:1–14:64, 2011.

[22] Dimitri Belli, Paolo Barsocchi, and Filippo Palumbo. Connectivity Standards Alliance Matter: State of the art and opportunities. *Internet of Things*, 25:101005, 2024.

[23] Lorenzo Bettini, Mario Coppo, Loris D'Antoni, Marco De Luca, Mariangiola Dezani-Ciancaglini, and Nobuko Yoshida. Global progress in dynamically interleaved multiparty sessions. In Franck van Breugel and Marsha Chechik, editors, *CONCUR 2008 - Concurrency Theory, 19th International Conference, CONCUR 2008, Toronto, Canada, August 19-22, 2008. Proceedings*, volume 5201 of *Lecture Notes in Computer Science*, pages 418–433. Springer, 2008.

[24] Adel Binbusayyis and Thavavel Vaiyapuri. Unsupervised deep learning approach for network intrusion detection combining convolutional autoencoder and one-class svm. *Applied Intelligence*, 51(10):7094–7108, 2021.

[25] Laura Bocchi, Tzu-Chun Chen, Romain Demangeon, Kohei Honda, and Nobuko Yoshida. Monitoring networks through multiparty session types. *Theor. Comput. Sci.*, 669:33–58, 2017.

[26] Grady Booch, James E. Rumbaugh, and Ivar Jacobson. The unified modeling language user guide. *J. Database Manag.*, 10(4):51–52, 1999.

[27] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: The insecurity of 802.11. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 180–189, 2001.

[28] AJ Bernheim Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. Home automation in the wild: challenges and opportunities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2115–2124, 2011.

[29] Marco Casagrande, Riccardo Cestaro, Eleonora Losiouk, Mauro Conti, and Daniele Antonioli. E-Spoofer: Attacking and Defending Xiaomi Electric Scooter Ecosystem. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2023.

[30] chadillac. MDNS Recon. https://github.com/chadillac/mdns_recon.git, 2013.

[31] Sara Chennoufi, Gregory Blanc, Houda Jmila, and Christophe Kiennert. Survey on intrusion detection systems in 5g. In *Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information (RESSI)*, 2023.

[32] Cognosec. ZIGBEE EXPLOITED The good, the bad and the ugly. https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly-wp.pdf, 2015.

[33] Christian Colombo and Yliès Falcone. Organising LTL monitors over distributed systems with a global clock. *Formal Methods Syst. Des.*, 49(1-2):109–158, 2016.

[34] Connectivity Standards Alliance (CSA). Matter Security and Privacy Foundamentals. https://csa-iot.org/wp-content/uploads/2022/03/Matter_Security_and_Privacy_WP_March-2022.pdf, 2022.

[35] Connectivity Standards Alliance (CSA). Matter v1.0 GitHub release. https://github.com/project-chip/connectedhomeip/releases/tag/v1.0.0, 2022.

[36] Connectivity Standards Alliance (CSA). Matter Official Documentation. https://project-chip.github.io/connectedhomeip-doc/index.html, 2024.

[37] Connectivity Standards Alliance (CSA). Matter Official SDK (connectedhomeip). https://github.com/project-chip/connectedhomeip, 2024.

[38] Connectivity Standards Alliance (CSA). Matter Specifications 1.4. https://csa-iot.org/developer-resource/specifications-download-request/, 2024.

[39] Connectivity Standards Alliance (CSA). Matter: The Foundation for Connected Things. https://csa-iot.org/all-solutions/matter/, 2024.

[40] Schutzwerk DE. Security Considerations for Matter Developers. https://www.schutzwerk.com/en/blog/matter-security-considerations/, 2023.

[41] Tamara Denning, Tadayoshi Kohno, and Henry M Levy. Computer security and the modern home. *Communications of the ACM*, 56(1):94–103, 2013.

[42] Daniel Dinu and Ilya Kizhvatov. EM analysis in the IoT context: Lessons learned from an attack on Thread. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 73–97, 2018.

[43] Mark Dowd. Malwairdrop: compromising idevices via airdrop. *Ruxcon, October*, 2015.

[44] Antoine El-Hokayem and Yliès Falcone. On the monitoring of decentralized specifications: Semantics, properties, analysis, and simulation. *ACM Trans. Softw. Eng. Methodol.*, 29(1):1:1–1:57, 2020.

[45] Daniel Fährmann, Laura Martín, Luis Sánchez, and Naser Damer. Anomaly detection in smart environments: a comprehensive survey. *IEEE access*, 2024.

[46] Yliès Falcone, Tom Cornebize, and Jean-Claude Fernandez. Efficient and generalized decentralized monitoring of regular languages. In Erika Ábrahám and Catuscia Palamidessi, editors, *Formal Techniques for Distributed Objects, Components, and Systems - 34th IFIP WG 6.1 International Conference, FORTE 2014, Held as Part of the 9th International Federated Conference on Distributed Computing Techniques, DisCoTec 2014, Berlin, Germany, June 3-5, 2014. Proceedings*, volume 8461 of *Lecture Notes in Computer Science*, pages 66–83. Springer, 2014.

[47] Yliès Falcone, Srdan Krstic, Giles Reger, and Dmitriy Traytel. A taxonomy for classifying runtime verification tools. *Int. J. Softw. Tools Technol. Transf.*, 23(2):255–284, 2021.

[48] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. Security analysis of emerging smart home applications. In *2016 IEEE symposium on security and privacy (SP)*, pages 636–654. IEEE, 2016.

[49] Earlence Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, and Atul Prakash. FlowFence: Practical data protection for emerging IoT application frameworks. In *25th USENIX security symposium (USENIX Security 16)*, pages 531–548, 2016.

[50] Colin J. Fidge. Partial orders for parallel debugging. In Richard L. Wexelblat, editor, *Proceedings of the ACM SIGPLAN and SIGOPS Workshop on Parallel and Distributed Debugging, University of Wisconsin, Madison, Wisconsin, USA, May 5-6, 1988*, pages 183–194. ACM, 1988.

[51] Behrang Fouladi and Sahand Ghanoun. Honey, I'm home!! - Hacking Z-Wave Home Automation Systems. https://sensepost.com/blog/2013/honey-im-home-hacking-z-wave-other-black-hat-news/, 2013.

[52] Ritam Ganguly and Borzoo Bonakdarpour. Stream-based decentralized runtime verification. *CoRR*, abs/2301.13266, 2023.

[53] Ritam Ganguly, Anik Momtaz, and Borzoo Bonakdarpour. Runtime verification of partially-synchronous distributed system. *Formal Methods Syst. Des.*, 64(1):146–177, 2024.

[54] GNUcitizen. Name (mDNS) Poisoning Attacks Inside The LAN. https://www.gnucitizen.org/blog/name-mdns-poisoning-attacks-inside-the-lan/, 2008.

[55] Google. Manage Your Smart Home With Google Home. https://home.google.com/welcome/, 2025.

[56] Google. OpenThread: an open-source implementation of the Thread networking protocol. https://github.com/openthread/openthread, 2025.

[57] Weijia He, Maximilian Golla, Roshni Padhi, Jordan Ofek, Markus Dürmuth, Earlence Fernandes, and Blase Ur. Rethinking access control and authentication for the home internet of things (IoT). In *27th USENIX Security Symposium (USENIX Security 18)*, pages 255–272, 2018.

[58] Ryan Heartfield, George Loukas, Anatolij Bezemskij, and Emmanouil Panaousis. Self-configurable cyber-physical intrusion detection for smart homes using reinforcement learning. *IEEE Transactions on Information Forensics and Security*, 16:1720–1735, 2020.

[59] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. In George C. Necula and Philip Wadler, editors, *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*, pages 273–284. ACM, 2008.

[60] Luke Houben, Thijs Terhoeve, and Savio Sciancalepore. MUDThread: Securing Constrained IoT Networks via Manufacturer Usage Descriptions. *IEEE Communications Magazine*, 2024.

[61] Danny Yuxing Huang, Noah Apthorpe, Frank Li, Gunes Acar, and Nick Feamster. Iot inspector: Crowdsourcing labeled network traffic from smart home devices at scale. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(2):1–21, 2020.

[62] Stephen S Intille. Designing a home of the future. *IEEE pervasive computing*, 1(2):76–82, 2002.

[63] Yan Jia, Bin Yuan, Luyi Xing, Dongfang Zhao, Yifan Zhang, XiaoFeng Wang, Yijing Liu, Kaimin Zheng, Peyton Crnjak, Yuqing Zhang, et al. Who's in control? on security risks of disjointed iot device management channels. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1289–1305, 2021.

[64] George Jiroveanu and René K. Boel. From local to global consistency in distributed monitoring of petri net models. *IEEE Trans. Autom. Control.*, 68(1):494–501, 2023.

[65] Julie A Kientz, Shwetak N Patel, Brian Jones, ED Price, Elizabeth D Mynatt, and Gregory D Abowd. The Georgia Tech Aware Home. *CHI'08 extended abstracts on Human factors in computing systems*, pages 3675–3680, 2008.

[66] Taehoon Kim and Wooguil Pak. Hybrid classification for high-speed and high-accuracy network intrusion detection system. *IEEE Access*, 9:83806–83817, 2021.

[67] Felix Klement, Emily Vorderwülbeke, and Stefan Katzenbeisser. One Standard to Rule Them All? Assessing the Disruptive Potential of Jamming Attacks on Matter Networks. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2023.

[68] Tiiu Koskela and Kaisa Väänänen-Vainio-Mattila. Evolution towards smart home environments: empirical evaluation of three user interfaces. *Personal and ubiquitous computing*, 8:234–240, 2004.

[69] Ingolf H. Krüger, Michael Meisinger, and Massimiliano Menarini. Interaction-based runtime verification for systems of systems integration. *J. Log. Comput.*, 20(3):725–742, 2010.

[70] Deepak Kumar, Kelly Shen, Benton Case, Deepali Garg, Galina Alperovich, Dmitry Kuznetsov, Rajarshi Gupta, and Zakir Durumeric. All things considered: an analysis of {IoT} devices on home networks. In *28th USENIX security symposium (USENIX Security 19)*, pages 1169–1185, 2019.

[71] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[72] Song Liao, Jingwen Yan, and Long Cheng. WIP: Hidden Hub Eavesdropping Attack in Matter-enabled Smart Home Systems. In *Workshop on Security and Privacy in Standardized IoT (SDIoTSec)*, 2024.

[73] Melissa Loos. Security analysis of the Matter protocol, 2023.

[74] Xiaoyue Ma, Lannan Luo, and Qiang Zeng. From One Thousand Pages of Specification to Unveiling Hidden Bugs: Large Language Model Assisted Fuzzing of Matter IoT Devices. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4783–4800, 2024.

[75] Basim Mahbooba, Mohan Timilsina, Radhya Sahal, and Martin Serrano. Explainable artificial intelligence (xai) to enhance trust management in intrusion detection systems using decision tree model. *Complexity*, 2021(1):6634811, 2021.

[76] Erwan Mahe, Boutheina Bannour, Christophe Gaston, and Pascale Le Gall. Efficient interaction-based offline runtime verification of distributed systems with lifeline removal. *Sci. Comput. Program.*, 241:103230, 2025.

[77] Erwan Mahe, Boutheina Bannour, Christophe Gaston, Arnault Lapitre, and Pascale Le Gall. Tooling offline runtime verification against interaction models: recognizing sliced behaviors using parameterized simulation. *J. Object Technol.*, 23(2):2, 2024.

[78] Erwan Mahe, Christophe Gaston, and Pascale Le Gall. Revisiting semantics of interactions for trace validity analysis. In Heike Wehrheim and Jordi Cabot, editors, *Fundamental Approaches to Software Engineering - 23rd International Conference, FASE 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings*, volume 12076 of *Lecture Notes in Computer Science*, pages 482–501. Springer, 2020.

[79] Erwan Mahe, Christophe Gaston, and Pascale Le Gall. Denotational and operational semantics for interaction languages: Application to trace analysis. *Sci. Comput. Program.*, 232:103034, 2024.

[80] Anna Maria Mandalari, Hamed Haddadi, Daniel J Dubois, and David Choffnes. Protected or porous: a comparative analysis of threat detection capability of iot safeguards. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 3061–3078. IEEE, 2023.

[81] Ravindra Mangar, Jingyu Qian, Wondimu Zegeye, Abdulrahman AlRabah, Ben Civjan, Shalni Sundram, Sam Yuan, Carl A Gunter, Mounib Khanafer, Kevin Kornegay, et al. Designing and Evaluating a Testbed for the Matter Protocol: Insights into User Experience. In *Workshop on Security and Privacy in Standardized IoT (SDIoTSec)*, 2024.

[82] Friedemann Mattern et al. *Virtual time and global states of distributed systems*. Univ., Department of Computer Science, 1988.

[83] Sjouke Mauw and Michel A. Reniers. Operational semantics for msc'96. *Comput. Networks*, 31(17):1785–1799, 1999.

[84] Menna Mostafa and Borzoo Bonakdarpour. Decentralized runtime verification of LTL specifications in distributed systems. In *2015 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2015, Hyderabad, India, May 25-29, 2015*, pages 494–503. IEEE Computer Society, 2015.

[85] Michael C Mozer. Lessons from an adaptive home. *Smart environments: Technologies, protocols, and applications*, pages 271–294, 2004.

[86] Faisal Nabi and Xujuan Zhou. Enhancing intrusion detection systems through dimensionality reduction: A comparative study of machine learning techniques for cyber security. *Cyber Security and Applications*, 2:100033, 2024.

[87] Subash Neupane, Jesse Ables, William Anderson, Sudip Mittal, Shahram Rahimi, Ioana Banicescu, and Maria Seale. Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities. *IEEE Access*, 10:112392–112415, 2022.

[88] Huu Nghia Nguyen, Pascal Poizat, and Fatiha Zaïdi. Passive conformance testing of service choreographies. In Sascha Ossowski and Paola Lecca, editors, *Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012*, pages 1528–1535. ACM, 2012.

[89] K Nimmy, M Dilraj, Sriram Sankaran, and Krishnashree Achuthan. Leveraging power consumption for anomaly detection on iot devices in smart homes. *Journal of Ambient Intelligence and Humanized Computing*, 14(10):14045–14056, 2023.

[90] G. Pickett. Port Scanning Without Sending Packets. `https://defcon.org/images/defcon-19/dc-19-presentations/Pickett/DEFCON-19-Pickett-Port-Scanning-Without-Packets.pdf`, 2011.

[91] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 46–57. IEEE Computer Society, 1977.

[92] Hassan Qadeer, Ammad Talat, Kashif Naseer Qureshi, Faisal Bashir, and Najam Ul Islam. Towards an efficient intrusion detection system for high speed networks. In *2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pages 428–433. IEEE, 2020.

[93] Xiaofei Qu, Lin Yang, Kai Guo, Linru Ma, Meng Sun, Mingxing Ke, and Mu Li. A survey on the development of self-organizing maps for unsupervised intrusion detection. *Mobile networks and applications*, 26:808–829, 2021.

[94] Sowmya Ramapatruni, Sandeep Nair Narayanan, Sudip Mittal, Anupam Joshi, and Karuna Joshi. Anomaly detection models for smart home security. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 19–24. IEEE, 2019.

[95] Dave Randall. Living inside a smart home: A case study. In *Inside the smart home*, pages 227–246. Springer, 2003.

[96] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. IoT goes nuclear: Creating a ZigBee chain reaction. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 195–212. IEEE, 2017.

[97] Grigore Rosu and Klaus Havelund. Rewriting-based techniques for runtime verification. *Autom. Softw. Eng.*, 12(2):151–197, 2005.

[98] Samsung. Samsung Smart Home with SmartThings. `https://www.samsung.com/us/smartthings/`, 2025.

[99] Arindam Sarkar, Hanjabam Saratchandra Sharma, and Moirangthem Marjit Singh. A supervised machine learning-based solution for efficient network intrusion detection using ensemble learning based on hyperparameter optimization. *International Journal of Information Technology*, 15(1):423–434, 2023.

[100] Alper Sen and Vijay K. Garg. Formal verification of simulation traces using computation slicing. *IEEE Trans. Computers*, 56(4):511–527, 2007.

[101] Koushik Sen, Abhay Vardhan, Gul Agha, and Grigore Rosu. Efficient decentralized monitoring of safety in distributed systems. In Anthony Finkelstein, Jacky Estublier, and David S. Rosenblum, editors, *26th International Conference on Software Engineering (ICSE 2004), 23-28 May 2004, Edinburgh, United Kingdom*, pages 418–427. IEEE Computer Society, 2004.

[102] So Seng, Joaquin Garcia-Alfaro, and Youssef Laarouchi. Why anomaly-based intrusion detection systems have not yet conquered the industrial market? In *International Symposium on Foundations and Practice of Security*, pages 341–354. Springer, 2021.

[103] Wooseok Seo and Wooguil Pak. Real-time network intrusion prevention system based on hybrid machine learning. *IEEE Access*, 9:46386–46397, 2021.

[104] Narmeen Shafqat and Aanjhan Ranganathan. Seamlessly Insecure: Uncovering Outsider Access Risks in AiDot-Controlled Matter Devices. In *2024 IEEE Security and Privacy Workshops (SPW)*, pages 281–288. IEEE, 2024.

[105] Kumar Shashwat, Francis Hahn, Xinming Ou, and Anoop Singhal. Security Analysis of Trust on the Controller in the Matter Protocol Specification. In *2023 IEEE Conference on Communications and Network Security (CNS)*, pages 1–6. IEEE, 2023.

[106] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. A billion open interfaces for eve and mallory: MitM, DoS, and tracking attacks on iOS and macOS through Apple Wireless Direct Link. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 37–54, 2019.

[107] Blase Ur, Jaeyeon Jung, and Stuart Schechter. The current state of access control for smart devices in homes. In *Workshop on home usable privacy and security (HUPS)*, volume 29, pages 209–218, 2013.

[108] Mathy Vanhoef and Frank Piessens. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1313–1328, 2017.

[109] Mathy Vanhoef and Eyal Ronen. Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 517–533. IEEE, 2020.

[110] Christopher Vattheuer, Charlie Liu, Ali Abedi, and Omid Abari. Are Home Security Systems Reliable? *arXiv preprint arXiv:2301.07202*, 2023.

[111] Eduardo Viegas, Altair Santin, Alysson Bessani, and Nuno Neves. Bigflow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. *Future Generation Computer Systems*, 93:473–485, 2019.

[112] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl Gunter. Fear and logging in the internet of things. In *Network and Distributed Systems Symposium*, 2018.

[113] Allison Woodruff, Sally Augustin, and Brooke Foucault. Sabbath day home automation: " it's like mixing technology and religion". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 527–536, 2007.

[114] Binghao Yan and Guodong Han. Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system. *IEEE Access*, 6:41238–41248, 2018.

[115] Igor Zavalyshyn, Axel Legay, Annanda Rath, and Etienne Rivière. SoK: Privacy-enhancing smart home hubs. *Proceedings on Privacy Enhancing Technologies*, 2022.

[116] Wei Zhou, Yan Jia, Yao Yao, Lipeng Zhu, Le Guan, Yuhang Mao, Peng Liu, and Yuqing Zhang. Discovering and understanding the security hazards in the interactions between IoT devices, mobile apps, and clouds on smart home platforms. In *28th USENIX security symposium (USENIX security 19)*, pages 1133–1150, 2019.