future internet

Review

A Holistic Analysis of Internet of Things (IoT) Security: Principles, Practices, and New Perspectives

Mahmud Hossain [1,*], Golam Kayas [2], Ragib Hasan [3], Anthony Skjellum [4], Shahid Noor [5] and S. M. Riazul Islam [6,*]

1 Department of Private Certificate Authority, Amazon Web Services (AWS), Herndon, VA 20171, USA
2 Department of Cybersecurity & Privacy Group, Comcast Crop., Philadelphia, PA 19103, USA; golam_kayas@comcast.com
3 Department of Computer Science, University of Alabama at Birmingham, Birmingham, AL 35294, USA; ragib@uab.edu
4 Department of Computer Science and Engineering, University of Tennessee at Chattanooga, Chattanooga, TN 37403, USA; tony-skjellum@utc.edu
5 Department of Computer Science, Northern Kentucky University, Highland Heights, KY 41099, USA; noors2@nku.edu
6 School of Natural and Computing Sciences, University of Aberdeen, Aberdeen AB24 3FX, UK
* Correspondence: mahmudhn@amazon.com (M.H.); riazul.islam@abdn.ac.uk (S.M.R.I.)

Abstract: Driven by the rapid escalation of its utilization, as well as ramping commercialization, Internet of Things (IoT) devices increasingly face security threats. Apart from denial of service, privacy, and safety concerns, compromised devices can be used as enablers for committing a variety of crime and e-crime. Despite ongoing research and study, there remains a significant gap in the thorough analysis of security challenges, feasible solutions, and open secure problems for IoT. To bridge this gap, we provide a comprehensive overview of the state of the art in IoT security with a critical investigation-based approach. This includes a detailed analysis of vulnerabilities in IoT-based systems and potential attacks. We present a holistic review of the security properties required to be adopted by IoT devices, applications, and services to mitigate IoT vulnerabilities and, thus, successful attacks. Moreover, we identify challenges to the design of security protocols for IoT systems in which constituent devices vary markedly in capability (such as storage, computation speed, hardware architecture, and communication interfaces). Next, we review existing research and feasible solutions for IoT security. We highlight a set of open problems not yet addressed among existing security solutions. We provide a set of new perspectives for future research on such issues including secure service discovery, on-device credential security, and network anomaly detection. We also provide directions for designing a forensic investigation framework for IoT infrastructures to inspect relevant criminal cases, execute a cyber forensic process, and determine the facts about a given incident. This framework offers a means to better capture information on successful attacks as part of a feedback mechanism to thwart future vulnerabilities and threats. This systematic holistic review will both inform on current challenges in IoT security and ideally motivate their future resolution.

Keywords: Internet of Things; analysis; security; communication security; device security; service security; forensic; threats; vulnerabilities; requirements; challenges; solutions; new perspectives

check for updates

Citation: Hossain, M.; Kayas, G.; Hasan, R.; Skjelum, A.; Noor, S.; Islam, S.M.R. A Holistic Analysis of Internet of Things (IoT) Security: Principles, Practices, and New Perspectives. *Future Internet* **2024**, *16*, 40. https://doi.org/10.3390/fi16020040

Academic Editor: Xuebin Ren

Received: 4 December 2023
Revised: 12 January 2024
Accepted: 17 January 2024
Published: 24 January 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With its manifold technical and functional benefits, the Internet of Things (IoT) has emerged as a significant paradigm to advance the Fourth Industrial Revolution [1–4]. The size and nature of the IoT continues to grow, and it will eventually be massive and pervasive. It enables an extensive set of applications and services such as home automation, environmental monitoring, healthcare, transportation, agricultural automaton, connected vehicles, energy efficiency and smart grid, remote monitoring, security, and safety. To do so, IoT devices make their services and data accessible to stakeholders, including end users and cloud services, through Internet connectivity. These services and data must be organized so that any form of access to the data is secure and limited to the parties involved. The security aspect of IoT emerges as the biggest concern; as the IoT spreads widely, digital mischief is likely to become a progressively physical threat [5–8].

To comprehend the significance of IoT security, one can investigate the existing state of IoT devices that are currently in operation. It is reported that 80% of devices exposed users' private information, such as name and date of birth, according to an assessment survey on commercialized IoT devices conducted by Hewlett-Packard (HP) [9]. The survey also found that 70% of the devices surveyed did not apply any encryption during communication, and 80% of them used fragile passwords in terms of adequate complexity and length. Moreover, of the devices, 60% had come with various security vulnerabilities, including cross-site scripting (XSS) [10] in their interfaces. A summary of the survey findings is provided in Figure 1. A number of studies have demonstrated that it is possible to efficiently establish control over the operation of commercialized IoT devices and perform various illicit operations. Some of those findings are noted in Table 1.

Table 1. Real attacks on smart systems.

Target Device	**Security Issue**	**Highlight**
Multimedia centers and appliances	Insecure Web interface	Compromised devices were used for sending phishing emails, and texts were sent from compromised Blu-ray devices and refrigerators [11].
Surveillance camera	Weak credentials	Internet-connected cameras were compromised to create a botnet and perform DDoS attacks on websites [12].
Programmable Logic Controller (PLC)	Insecure firmware	Reprogrammed with rootkit [13]
	Insecure operating system	Processed malicious commands [14]
Webcams and smart bulbs	Account enumeration	A botnet was formed using a large number of compromised devices [15]
Thermostat	Lack of access control methods	A thermostat was compromised to shut down the heating of a building [16]
Vehicle	Insecure Controlled Area Network (CAN) interface	Adversaries took control over radio, dashboard, brake, and acceleration [17]
Drug pump	Insufficient authentication and authorization	Adversaries changed the dose of the drug pump [18]
Baby monitor	Insufficient authentication and authorization	An Internet-connected baby monitor allowed unauthorized access to its camera [19]
Traffic sensors	Insecure firmware	Adversaries sent fake data to traffic control systems [20]
Smart TV	Insecure communications	Adversaries eavesdropped on broadcast messages [21]

A Deeper Look at Why IoT security is different: The inherent heterogeneity in services [22], applications, and devices in IoT systems introduces multi-modal complexity to the IoT security solution architects and providers. The development of novel IoT security schemes faces further challenges because of the presence of diverse communication media and respective proprietary protocols. Moreover, the contemporary security schemes applied in laptops, personal computers, and smartphones cannot be directly adopted for IoT systems due to the resource-constrained nature of IoT sensors, devices, and networks. As reported in Table 2, IoT devices come with memory space of a few megabytes (typically, 8–32 KB of RAM and 48–512 KB of ROM) and a CPU with low power consumption (corresponding to a clock frequency of 8–96 MHz), and they have to work with a low-bandwidth connection (corresponding to a low data rate of 16–250 kbps) [23–27].

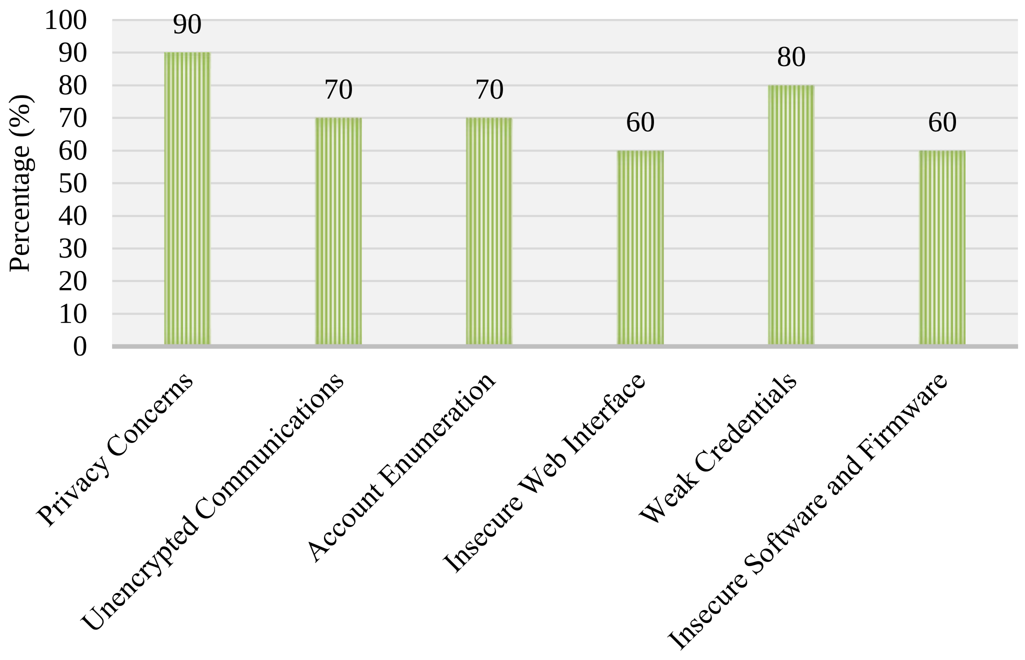


Figure 1. Security issues with commercialized IoT devices [9].

Table 2. Specifications of some representative devices used for IoT applications.

Device Specification	**CPU**		**Storage**		**Networking**		
	Arch (Bits)	**Clock (MHz)**	**RAM (KB)**	**ROM (KB)**	**Standard**	**Radio Interface**	**BW (kbps)**
Sky-Mote [24]	16	8	10	48	6LoWPAN	IEEE 802.15.4	250
Z1-Mote [23]	32	32	32	512	6LoWPAN	IEEE 802.15.4	250
Openmote [26]	32	32	32	512	6LoWPAN	IEEE 802.15.4	250
Waspmote [27]	8	16	8	128	Zigbee	IEEE 802.15.4	250
Arduino Uno [28]	8	16	2	32	6LoWPAN	IEEE 802.15.4	250
Mbed [29]	32	96	32	512	CAN	CAN Bus	320
Weptech [30]	32	32	32	512	6LoWPAN	IEEE 802.15.4	250
KNX Stacks [31]	32	32	32	512	KNX	KNX Radio	16.4

Some of the pivotal constraints on employing conventional security solutions in IoT systems are highlighted below:

1. IoT devices have low-powered CPUs, and most are battery powered. The cryptographic algorithms used in conventional security methods may not be executable on IoT devices, as these devices operate at a slower clock speed.
2. IoT devices have less memory and storage compared to conventional digital devices, such as smartphones and laptops. The security protocols used by conventional devices may not consider memory limitations in their design, so IoT devices may not have enough space in the RAM to load and execute the conventional security methods after loading embedded software, such as operating systems, services, and applications.
3. IoT devices communicate over low-data-rate radio interfaces. Conventional security methods may not be optimized for these lossy and low-powered communication links.

An IoT device may not respond to a real-time request if it spends most of its assigned time slots for serving a request on exchanging security messages.

4. IoT devices use lightweight operating systems, such as Contiki [32] and RIoT [33], due to their resource-constrained natures. As such, the protocol stack of IoT operating systems requires a resource-efficient version of contemporary security modules, such as IPsec [34] and DTLS [35], to run on IoT devices.
5. IoT software has to be updated regularly to mitigate potential security vulnerabilities. However, the real-time and lightweight operating systems that run on IoT devices may not have the capability to receive and integrate new codes or libraries to keep the system software updated.
6. IoT networks are expected to experience abrupt changes in network topologies because mobile IoT devices may join a network without prior configuration or leave the network abruptly. The sudden changes in the network topologies may affect various performances of the existing security methods, such as re-distribution of shared credentials in the pre-shared key-based authentication methods. As a result, conventional security schemes may not be suitable for mobile IoT-based systems.
7. A wide range of wireless protocols are used for communications in the IoT systems, which include WiFi [36], ZigBee [37], Z-Wave [38], and NFC [39]. A smart device can use proprietary networking protocols for device-to-device communications and standard protocols for Internet communications. The conventional security methods may not be comprehensive enough for the entire set of properties of each communication protocol.

The above constraints suggest that the utilization of IoT-based systems without appropriate security measures puts the success of the evolving IoT paradigm at risk, which would downgrade or even destroy the whole IoT business. We must be careful to investigate and determine priorities for IoT security solutions. Accordingly, security solutions and guidlines used in designing IoT elements, systems, and protocols should be customized for the context of use. In this paper, we survey contemporary IoT security issues and perform an in-depth critical analysis of them with the aim of educating learners and practitioners in IoT security implications.

1.1. Existing Surveys and Our Contributions

From a general viewpoint, some of the security and privacy issues in IoT have been discussed in [40–45]. However, these papers primarily survey IoT progress in general covering applications, enabling technologies, architectures, and some security fundamentals. Some of the IoT-related security concerns have been surveyed in Refs.[46,47] provides a survey on M2M research security, and [47] discusses the challenges and solutions for securing fog computing for IoT applications. In regard to IoT-enabled cyberattacks, the authors in [48] deliver an assessment survey on attack paths to relevant infrastructures and services. Ref. [49] presents a reasonable review of intrusion detection systems for IoT technologies, but the paper mainly focuses on architecture types.

Recently, a few reasonable surveys on IoT security [50–59] have been made available in the literature; however, the spectrum of IoT security is so large that many critical issues and aspects are yet to be investigated. In Table 3, we show a comparative analysis on the scope of our paper with that of the existing surveys. Mrabet et al. [59] divided the IoT attacks into five different layered architectures and presents security issues of each layer with the security measures and mechanisms to defeat these attacks. However, a more useful classification of the prominent IoT attacks based on other parameters, such as devices' properties and severity level, may help us to better understand and analyze the exiting security problems and mitigation techniques. Similarly, Mohanta et al. [50] presented different attacks taxonomy based on the different layer of IoT architecture and discusses the use of emerging technologies, such as artificial intelligence and blockchain in mitigating these attacks. Nevertheless, a more comprehensive discussion on IoT attack surface and vulnerabilities of IoT ecosystem is desired. Ahmed et al. [60] primarily explored the integration of Blockchain and IoT, specifically focusing on energy, security, and hardware aspects. This provided insights into the challenges and opportunities associated with this integration, offering a specialized examination within the mentioned domains. Hewa et al. [61] primarily concentrated on bolstering security in cloud manufacturing equipment clusters for Industry 4.0. By leveraging edge-based blockchain and fog computing, they targeted challenges related to privacy, authentication, and overall system performance. In contrast, our work takes a broader approach, systematically analyzing security vulnerabilities across IoT layers and proposing a comprehensive Blockchain-based forensic framework. While both works contribute significantly to IoT security, our research spans a wider spectrum, addressing diverse aspects beyond the specific focus of cloud manufacturing.

Table 3. Comparative analysis with the prior survey works.

Aspects		**[59] et al.**	**[50] et al.**	**[51] et al.**	**[52] et al.**	**[53] et al.**	**[54] et al.**	**[55] et al.**	**[56] et al.**	**[57] et al.**	**[58] et al.**	**Our Work**
Attack Surface & Vulnerability Identification	Device	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓
	Cryptography	✗	✗	✗	✓	✓	✓	✗	✗	✓	✓	✓
	Communication	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
	Service	✓	✗	✓	✓	✗	✗	✗	✓	✓	✗	✓
Attack Taxonomy	Adversary Location	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✓
	Device Property	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	✓
	Data Privacy	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓
	Severity	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
Security Requirements	Access Control	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
	Confidentiality	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓
	Availability	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
Protocol Stack-wise Security Solutions	Efficient Cryptography	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓
	Device	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
	Transport Layer	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✓
	Network Layer	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓
	Application Layer	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✓
	Access Level	✗	✗	✗	✗	✓	✓	✓	✗	✓	✓	✓

Another recent work [51] demonstrated the feasibility of using deep-learning-based techniques and provides a taxonomy of the machine-learning-based methods for IoT security. The research also illustrates the common IoT security threats and the attacks surfaces of an IoT ecosystem focusing on the use of machine learning and artificial intelligence. However, the paper does not cover the detailed security requirements needed to ensure secure operational and access model, a very much desirable aspect in IoT security. Then, Stoyanova et al. [52] focused on fundamental challenges, theoretical frameworks, and research trends in the IoT forensics. The authors discuss existing IoT forensics and their usefulness and provide some guidelines to the digital forensics professionals. The research provides a good taxonomy of the attacks on IoT systems, but does not focus on the prospective solutions to mitigate the attacks. S-FoS [62] proposes an SDN-based security-aware workflow scheduler for IoT-Fog networks, defending against DDoS and port scanning attacks by integrating fuzzy-based anomaly detection and NSGA-III multi-objective scheduling optimization. Through simulations, it outperforms existing algorithms, improving response time and network utilization in varying IoT scenarios. Unlike S-FoS, our paper focuses on the broader spectrum of IoT security, addressing vulnerabilities across various layers, communication protocols, and network topologies. We provide a comprehensive analysis, categorizing threats and proposing security solutions tailored to the unique constraints of IoT devices, emphasizing the need for customized security measures in the evolving IoT paradigm. In prior research, the authors proposed FUPE [63], a security-aware task scheduler for IoT-Fog networks, leveraging Software-Defined Networking (SDN) to address TCP SYN flood attacks. FUPE employs a fuzzy-based multi-objective particle swarm optimization approach to optimize computing resources and enhance security simultaneously. Extensive simulations demonstrate FUPE's superiority over state-of-the-art algorithms, showcasing significant improvements in average response time and network utilization under varying attack rates, fog devices, and job numbers. While FUPE focuses on addressing TCP SYN flood attacks in IoT-Fog networks through a security-aware task scheduler with SDN and optimization techniques, our work takes a broader perspective on IoT security. Our paper provides a comprehensive analysis of various security threats and vulnerabilities in IoT, presenting a detailed attack taxonomy. We categorize threats across different layers, communication protocols, and network topologies. Additionally, our proposed security solutions cover diverse aspects, offering a holistic approach to IoT security challenges.

Javanmardi et al. [64] explored the security challenges in IoT-Fog networks, emphasizing the vulnerabilities and attacks in the fog layer. They proposed a Blockchain and Fog-computing-enabled security service architecture, utilizing Hyperledger Fabric for fog nodes at the edge. The focus is on authentication, equipment-cloud channel privacy, and defense against malicious attacks. This work primarily addressed energy efficiency and hardware aspects of IoT and Blockchain integration, while our work takes a broader approach. We comprehensively analyze various security threats and vulnerabilities in IoT, offering a detailed attack taxonomy. Our proposed security solutions cover diverse layers, communication protocols, and network topologies, providing a holistic view of IoT security. The work of Lounis et al. [53] presented the attacks and the mitigation techniques related to the wireless infrastructures of IoT systems covering the most frequently used wireless communication technologies from the resource-constrained perspectives. The authors also provided a classification of the attacks based on a security-service-based attack. They also presented mitigation techniques of certain attacks, provided some guidelines to the user, and highlighted the limitation of these security measures. However, the complete security requirements of the IoT systems in different levels, such as operational level, information level, and access level, is not a part of the research. Similarly, the work of Sharma [54] focused on the security, privacy, and trust of mobile IoT (M-IoT) devices. The authors also discussed several available secure frameworks for M-IoT devices, such as an access control and authorization-based framework, risk-assessment-based framework, authentication-based framework, and secure services-based framework as a secure solution to tackle IoT vulnerabilities and attacks. In contrast, more rigorous classification of the attack surface and complete security of different level of the IoT network is not considered therein. Sha et al. [55] aimed to present the prior works of edge-based security designs for IoT solutions. The authors also proposed an edge-centric IoT architecture. However, the other important aspects of the IoT systems attack surfaces, such as service vulnerabilities and cryptographic weaknesses, were not discussed. Meneghello et al. [65] addressed the IoT security issues and probable counter measures from a more practical perspective. This article focused on four communication protocols mostly used in IoT devices, namely ZigBee, Bluetooth low energy (BLE), 6LoWPAN, and LoRaWAN. However, a general layer-wise security solution was not discussed in that research.

In this article, we present a holistic analysis of IoT security problems that includes a detailed discussion on the vulnerabilities and attack surface of the IoT network. We also provide a comprehensive attack taxonomy of the IoT attacks based on adversary and device properties, data privacy, and severity level. Moreover, the available effective techniques to secure the IoT solutions are also presented in this research. Furthermore, we also propose a Blockchain-based forensic framework to investigate and identify security issues of the IoT-based systems. Our specific contributions are highlighted as follows:

1. We provide a comprehensive analysis of various security threats and vulnerabilities in IoT. Different from the existing surveys, this article identifies various attack surfaces and categorically discusses the associated vulnerabilities. We formulate an attack tree to classify the attacks in terms of their severity level and present a detailed attack taxonomy that encompasses a wide spectrum of how devices, hosts, access levels, locations, and strategies, among others, play a role in initiating respective attacks.
2. From a systems design perspective, we introduce the concept of a security landscape that can reflect multi-modal complexity based on applications, services, devices, and connectivity associated with an IoT system of interest. Then, we examine properties required for various security schemes, including access level and functional security requirements.
3. With the aim of mitigating various threats, we classify the existing security solutions into three categories (end device security, communication security, and service security) and thoroughly discuss each of them. Subsequently, this article presents several comparative analyses of the proposed security schemes.
4. We find that the existing research primarily addresses the information- and access-level security properties. However, the time has come to pay attention to the resource efficiency and functional robustness of the security schemes. Accordingly, we identify open research problems and provide guidelines for future research directions.
5. It is common that smart devices, applications, and communications become a subject, object, or tool related to IoT crimes, and appropriate investigations should, therefore, be conducted to execute a cyber-forensic process and determine the facts behind attacks. With this perspective and in the context of an IoT-based system that might consists of billions of smart devices, we propose a Blockchain-based forensic framework. The framework can potentially assist a forensics investigator in defining evidence, developing scalable storage mechanisms to log a large amount of evidence, and generating secure provenance of the evidence.

1.2. Organization

A list of acronyms used throughout the paper is presented in Table 4. Figure 2 illustrates the organization of this manuscript. Section 2 describes the operational model of an IoT-based system. We identify security vulnerabilities in IoT-based systems in Section 3. The analysis of threat model with various security risks and attacks is presented in Section 4. Security requirements of an IoT-based system are presented in Section 5. Current solutions to mitigate IoT attacks are discussed in Section 6. Section 7 enumerates open research problems in the IoT environment. Finally, we conclude in Section 8.

Table 4. List of acronyms and corresponding definitions.

Acronyms	Definitions
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
API	Application Programming Interface
BLE	Bluetooth Low Energy
CAN	Controlled Area Network
CapBAC	Capability-based Access Control
CoAP	Constrained Application Protocol
DDOS	Denial of Service
DH	Diffie Hellman
DODAG	Destination Oriented Directed Acyclic Graph
DOS	Denial of Service
DTLS	Datagram Transport Layer Security
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie Hellman
HIP	Host Identity Protocol
IDS	Intrusion Detection System
IoV	Internet of Vehicle
IPS	Intrusion Prevension System
IPsec	Internet Protocol Security
LLN	Low Power and Lossy Netwroks
M2M	Machine-to-Machine
MTU	Maximum Transmission Unit
NFC	Near Field Communication
RPL	IPv6 Routing Protocol for LLNs
RSU	Road Side Unit
SAML	Security Assertion Markup Language
TLS	Transport Layer Security
V2V	Vehicle-to-Vehicle
XACML	eXtensible Access Control Markup Language
XSS	Cross Site Scripting

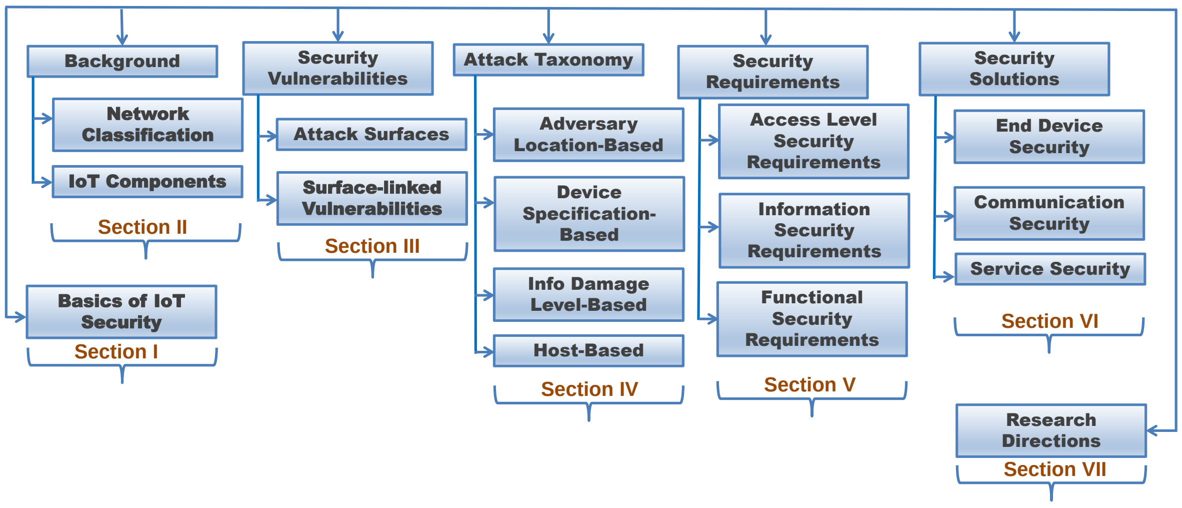


Figure 2. The organization and structure of this paper.

2. Background

We present components and operational model of an IoT system [66,67] in Figure 3. The details of the components are as follows.

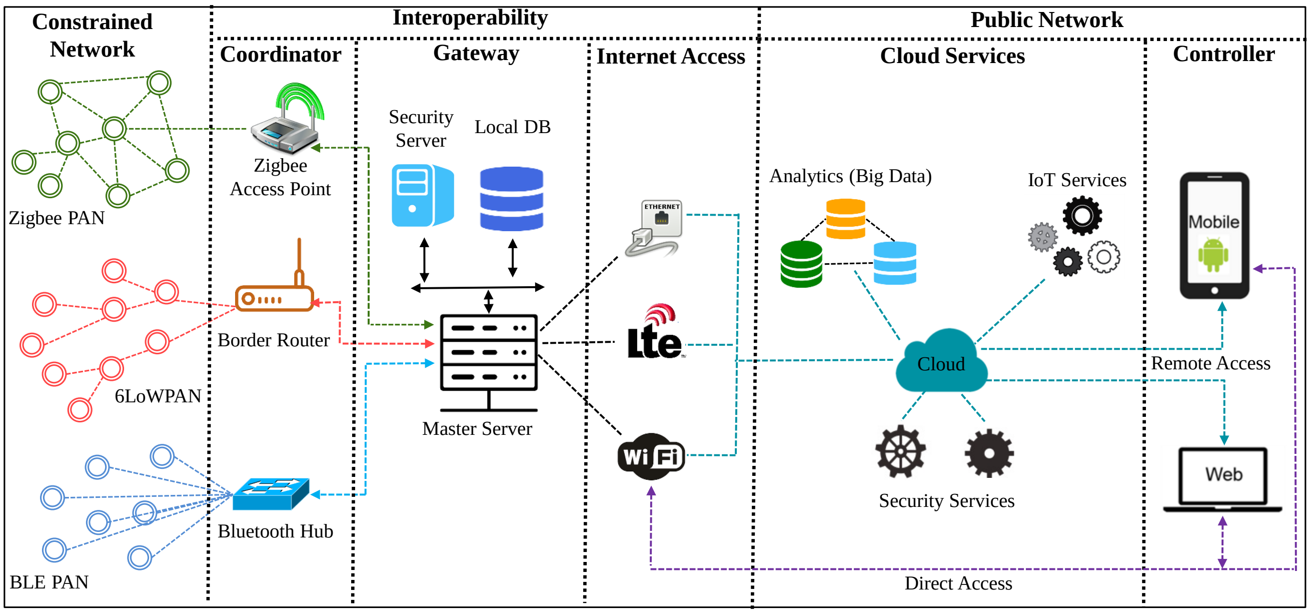


Figure 3. The operational model of an IoT system.

2.1. *IoT Networks*

An IoT system comprises two types of networks: constrained network and public network. The smart devices operate in the constrained network. As shown in Figure 3, in a constrained network, the devices can use multiple protocols, such as 6LoWPAN [68], Zigbee [69], ZWave [38], and BLE [70], for communications. The communication links of a constrained network are low power and lossy. Therefore, smart devices exchange information between them at a rate which is significantly lower than conventional digital networks, such as Ethernet and WiFi. In the constrained network, smart devices use IPv6 for addressing and CoAP [71] as the application protocol. In constrained networks, the RPL routing protocol [72] is used to route network packets over the lossy links.

The Internet (Wide Area Network) is considered to be the public network in an IoT system. A public network provides connections between multiple IoT systems, such as smart home, building, medical, and industry, located in the edge of the networks. The 3G, 4G, or 5G technologies can be used for communication in the public networks. The data rates of the pubic networks are significantly higher than those of the constrained networks. The public networks use IPv4 and HTTP for message delivery and conventional routing protocols [73], such as RIP and OSPF, for routing packets.

2.2. *IoT Device*

An IoT device is embedded with various software, hardware, and network components. The software component consists of operating systems, micro-services, and applications. The hardware component includes sensors, actuators, and batteries. A radio transceiver is embedded with a device as a network component.

An IoT device collects contextual information using its sensors. The device uses its actuators to perform various actions based on the collected information. The device may also perform an action based on the commands received from its owners and users.

To better understand the operating method of a smart device, we consider two scenarios of a smart home and healthcare system. In a smart home, a wearable device may adjust the temperate of an air conditioner according to the humidity of the room and perspiration level of the device owner. Similarly, in a smart healthcare system, medical sensors attached to a patient body may record the health conditions of the patient and send the information to hospitals and physicians for analysis.

2.3. IoT Service

IoT services can be classified into two categories: edge IoT service and cloud IoT service. The edge IoT services are located in the constrained networks and provided by the smart devices. Users interact with the smart devices using the edge services. Smart devices also exchange information with them using these services. The edge services may also be used for device maintenance purposes, such as updating or upgrading embedded software.

Cloud IoT services are located on the Internet and publicly accessible. These services enable users to interact with smart devices remotely. The cloud IoT services communicate to the edge services to receive sensor information and then process the information for making decisions. The cloud IoT services also handle the tasks of device registration, device management, sensor data management, and process automation.

2.4. Coordinator

A coordinator device can be considered as a device hub that manages a group of smart devices. Smart devices perform actions that trigger multiple events. A coordinator collects these actions and events for reporting purposes. The coordinator also monitors health, including battery percentage and resource utilization, of the smart devices that operate under it. The coordinator sends aggregated reports on the actions, events, and device health to the IoT service providers, device manufacturers, and system administrators. These reports can be used for audit or accounting purposeds. The information can also be used for big data analysis.

2.5. Gateway

An IoT Gateway enables communication between constrained and public networks. A smart device sends sensor information to a cloud IoT service through the Gateway. Similarly, cloud IoT services can communicate with a smart device through the Gateway to receive real-time updates. The Gateway acts as a protocol translator in an IoT network. As such, devices with various networking protocols can communicate with them. For instance, the Gateway allows communications between a Zigbee device and a 6LoWPAN device. A Zigbee device does not use IPv6 for addressing, while a 6LoWPAN device does. The Gateway receives a Zigbee packet, translates the packet to an IPv6 packet, and forwards it to the 6LoWPAN network. Hence, the Gateway enables cross-protocol communications in IoT networks. Moreover, the Gateway translates IPv6 and CoAP protocols to IPv4 and HTTP, respectively, to allow interactions between edge and cloud IoT services.

2.6. Controller

A controller device consumes edge and cloud IoT services. Device owners and service users use a controller entity, such as smartphone and web applications, to interact with smart devices. For instance, physicians can monitor patients' health conditions using their smartphones. A controller device can be co-located with a smart device in the constrained network to interact with the smart device. The controller device can also communicate with an IoT device remotely being located on the Internet.

3. Security Vulnerabilites

In this section, we first identity attack surfaces of an IoT system. Next, we provide details on the vulnerabilities associated with these attack surfaces.

3.1. Attack Surfaces

In an IoT system, the number of communication interfaces increases significantly compared to traditional digital systems because of the distribution of heterogeneous devices, communication media, networking protocols, services, and applications [74]. As a result, the number of attack vectors increases many folds in the IoT environment. As public IoT services are located in cloud servers, the attack surfaces from the cloud computing paradigm also contribute to the increment of the attack vectors [75]. Figure 4 shows various types of communication interfaces in an IoT system. As presented in Table 5, in the constrained network, adversaries can perform attacks by exploiting vulnerabilities in things-to-things, things-to-controllers, controllers-to-gateways, and gateways-to-users communication interfaces. Moreover, the vulnerabilities in gateways-to-clouds and clouds-to-users interfaces can be exploited to compromise IoT services.

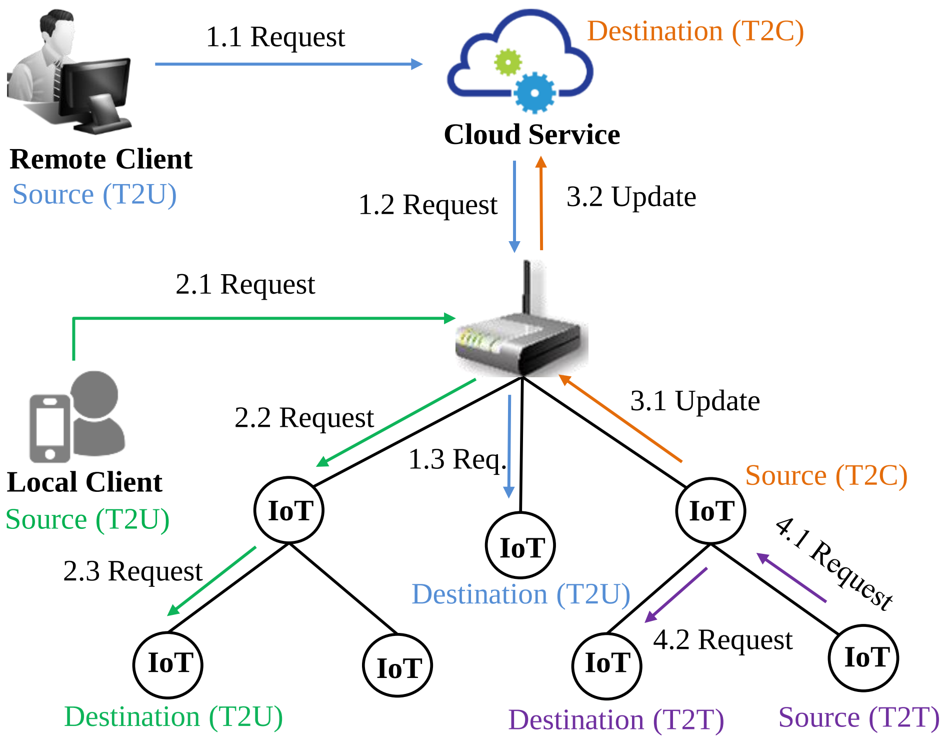


Figure 4. Communication interfaces in an IoT system.

Table 5. IoT attack surfaces.

Network	Attack Interface
Constrained Network	Things ⇔ Things Request 4 in Figure 4: a wearable medical sensor interacts with a thermostat to adjust room temperature.
	Things ⇔ Coordinator Zigbee network: the communication interface between Zigbee nodes and Zigbee Access Point as shown in Figure 3
	Things ⇔ Controller Request 2 in Figure 4: a user controls home appliances using a smartphone
Public Network	Things ⇔ Cloud Service Request 3 in Figure 4: the interfaces between IoT nodes and Gateway as well as between Gateway and Cloud
	Cloud Service ⇔ Controller Request 1 in Figure 4: a physician monitors a patient's medical devices remotely
	Cloud Service ⇔ Cloud Service Interfaces between a medical service managed by hospitals and medical data analysis service maintained third-party providers

3.2. Surface-Associated Vulnerabilities

In the following sections, we present various vulnerabilities that can be associated with the attack surfaces of smart systems [76,77].

3.2.1. End Device Vulnerability

Vulnerable Device Role: An IoT device can have multiple roles in a smart system. A device embedded with a sensor performs as a collector. Similarly, a device with a sensor and actuator collects information as well as performs actions. It is also possible that a device has a combined role of collector, performer, and coordinator. The multiple roles of IoT nodes can make them vulnerable to identity thefts and make it easy for a malicious device to impersonate a legitimate device [78].

System Software Vulnerability: IoT devices are embedded with system software, such as operating system, kernel, and firmware. The system software should be upgraded and patched regularly to avoid exploitations. However, due to the lossy nature of the constrained networks, it becomes a challenging task to apply security updates over the air to an IoT system with a large number of smart devices. As such, smart devices are at risk of being compromised if their software is not patched regularly. Moreover, the task of updating and upgrading should be performed by following proper guidelines. Otherwise, an insecure update may enable adversaries to discover security-critical information related to software and firmware versions and configurations [79,80].

Storage Vulnerability: A smart device stores security-sensitive information, such as cryptographic materials and sensor information, in its non-volatile storage. Device manufacturers may not consider storage security in the hardware design to reduce the size of a smart device or minimize the price of the device. Therefore, poor storage security can be a threat to the privacy and confidentiality of the in-device sensor information and credentials [81].

3.2.2. Communication Vulnerability

Vulnerable Multi-Protocol Connectivity: An IoT system comprises various types of networking protocols and communication links, including wireless, wired, intranets, and Internet [66,67]. A vulnerability in one networking protocol can be propagated to another protocol during cross-protocol communications. Moreover, the privacy and confidentiality of sensor information can be breached during the protocol translation performed by the Gateway. A Gateway device translates IPv6 packets to IPv4 packets when an IoT device exchange messages with a cloud server. A malicious Gateway device can learn about the payloads while performing the protocol translation. As a result, communication between the heterogeneous networks is vulnerable to various attacks that can be a threat to the violation of information privacy and confidentiality.

Network Service Vulnerability: Security features, such as Firewall, to configure network services with proper security settings may not be present in the IoT operating systems. The settings may include MAC address, IP address, and port filtering. For example, Contiki [32] and RIoT [33] operating systems do not have options to configure firewall settings. As such, ports that are used for device maintenance can be exposed to adversaries [82].

Vulnerable Cryptography: IoT devices may not adopt strong cryptographic schemes, such as larger keys, to encrypt communications because they have limited storage capacity, network bandwidth, and CPU power. Adversaries may find it easy to learn the communications as they are encrypted using weak encryption methods.

3.2.3. Service Vulnerability

Vulnerable Edge Service: IoT devices provide various types of services in the edge networks. These devices expose Web and Application Programming Interfaces (APIs) that enable users to interact with them. For instance, a smart thermostat allows a user to adjust the room temperature using their smartphone. A Coordinator device enables a network administrator to manage the IoT nodes through a device management service. These interfaces can be vulnerable to various web attacks, such as SQL injection [83], password enumeration [84], and cross-site scripting [85], if they are not designed with proper security guidelines. The services may not implement a security policy that locks a user's account after a limited number of password guesses or may support vulnerable words as account credentials, such a a dictionary word, as a password [76,77].

Vulnerable Cloud Service: Cloud IoT services enable users to interact with smart devices remotely. These services also store sensor information in the cloud servers for further analysis. Although the cloud services ensure a user's seamless access to IoT nodes, they introduce various security risks at the same time. For instance, there may have security threats to data privacy and confidentiality, service availability, and continuous access to IoT nodes if cloud servers and services are not patched with the latest security updates [86].

Vulnerable Partner Cloud: IoT applications and cloud services may interact with third party clouds to provide analytics to users. A smart healthcare system can share sensor information from a patient's medical devices with third-party cloud services to help physicians to understand the patient's health condition(s). Sharing the data with third-party clouds can lead to critical security risks, such as privacy and confidentiality threats, if the third-party clouds do not adopt proper security schemes to receive and share the shared information. A third-party cloud provider may not use a reliable security method for user authentication, message encryption, and message integrity verification. The third-party cloud service can use SHA-1, which is vulnerable to collision attacks [87], as the hash algorithm to verify message integrity.

4. Attack Taxonomy

In this section, we present a classification of IoT attacks. We classify the attacks based on the location of an adversary, the types of devices that can be used in attacks, level of information damage in attacks, and techniques used for compromising credentials.

4.1. Attacks Based on Adversary Location

External Attack: An attacking device is located anywhere on the Internet. Figure 5 shows the locations of the adversaries in the external networks. The threat model for external attacks can be as follows. A malicious application can be published in the application store, and later this application can be installed on a smartphone that is used to control IoT devices. Similarly, a malicious application can be installed on the cloud servers where sensitive sensor information is stored. The malicious applications can leak sensor data and users' personal information, such as a patient's health conditions received from their medical devices, to the Internet. Moreover, the malware can exploit vulnerabilities in the Gateway device [88,89] of an IoT network to gain unauthorized access to smart devices and networks remotely. The malware can perform various web service attacks, such as SQL injection and cross-cite scripting, using the compromised Gateway device.

Internal Attack: An attacking device and target devices are located in the same IoT network. The attacking device can be a member of the Destination-Oriented Directed Acyclic Graph (DODAG) [90] formed by smart devices, or it can be deployed to a location in the IoT network such that target devices and the attacking device share the same radio signal used in communications. Figure 6 shows the positions of an attacking device in an IoT network. In the internal attack, an attacking device is used for gathering information on communications protocols and network vulnerabilities. For instance, the attacking device can be used to overhear communications between IoT devices to determine whether messages are exchanged in plain text or are encrypted. If message are encrypted, then it can also be determined whether or not a vulnerable security method is used for encryption, such as heartbleed vulnerability in TLS 1.1 and 1.2 [91]. The attacking device can also be used to identity vulnerabilities in the software used by IoT devices, such as open ports. An adversary can use this information to gain unauthorized access to IoT devices or to perform various network attacks, such as wormhole [92], sinkhole [93], and botnet [94].

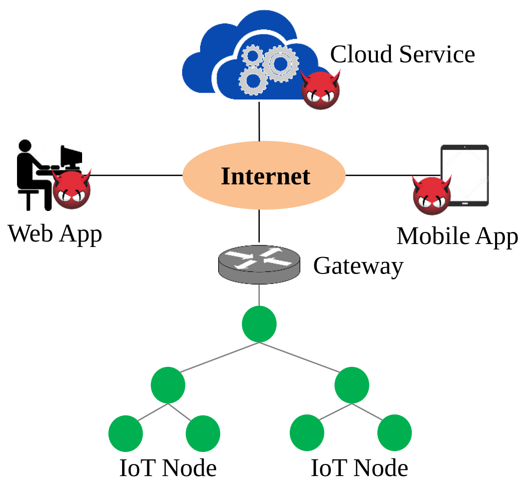


Figure 5. External attack.

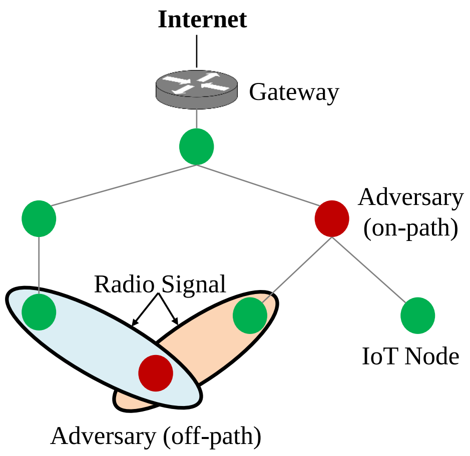


Figure 6. Internal attack.

4.2. Attacks Based on Device Property

High-End Device Class Attack: Malicious devices used to perform attacks have more resources—such as CPU speed, storage capacity, network bandwidth, and battery power—than a target device has. An attacker can use desktop, laptop, and cloud PCs to perform attacks on IoT networks and smart devices. The attacking device can be either co-located with the target device or positioned outside of the network of the target device. An adversary uses the high-end devices to perform attacks that increase the resource utilization of a victim device [95,96]. An attack scenario for a high-end device attack is as follows. An adversary sends a large number of malicious requests to the target device in a very short period. The target device keeps allocating resources to process the malicious requests and eventually suffers from resource exhaustion, such as memory overflow and reduced battery life. As a result, the target device cannot handle legitimate requests or send time-sensitive sensor information to its users due to resource unavailability. Although an attacker's goal is to stop a target device from providing services by exhausting its resources, the adversary uses a high-end device as an attacking device to avoid such resource exhaustion while performing attacks.

Low-End Device Class Attack: An IoT device is used as a tool for attacking another IoT device. As such, the attacking and target devices have similar capabilities and resource specifications regarding CPU, storage, and network bandwidth. In the low-end device class attacks, attacking devices positioned inside an IoT network are configured to perform Distributed Denial of Service (DDOS) type attacks. For example, an attacker can maliciously configure a smart thermostat to provide false information on room temperature to its owner. Similarly, a smart watch containing malware can introduce itself as a smart light. Next, the smart watch can exploit the trust relationship between the smart light and a smart TV to get unauthorized access to the TV. Later, the smart watch can use the TV as a platform for sending spam emails.

4.3. Attacks Based on Information Damage Level

Interruption: An adversary limits the capabilities of device-to-device and device-to-gateway communications by interfering with radio signals. The adversary can use certain jamming devices to interfere with radio signals [97]. Interruption attacks are threats to quality and availability of services provided by an IoT device.

Man-in-the-Middle: In the Man-in-the-Middle (MITM) attack, a malicious IoT node sits between two victim nodes and tricks them into thinking that they are communicating with each other. Figure 7 shows an example of the MITM attack in the RPL network. A malicious node M spoofs the identities of node A and B. Node M establishes a session with node A impersonating Node B and initiates another session with node B impersonating node A. Hence, Node M tricks Node A and Node B into providing sensor information.

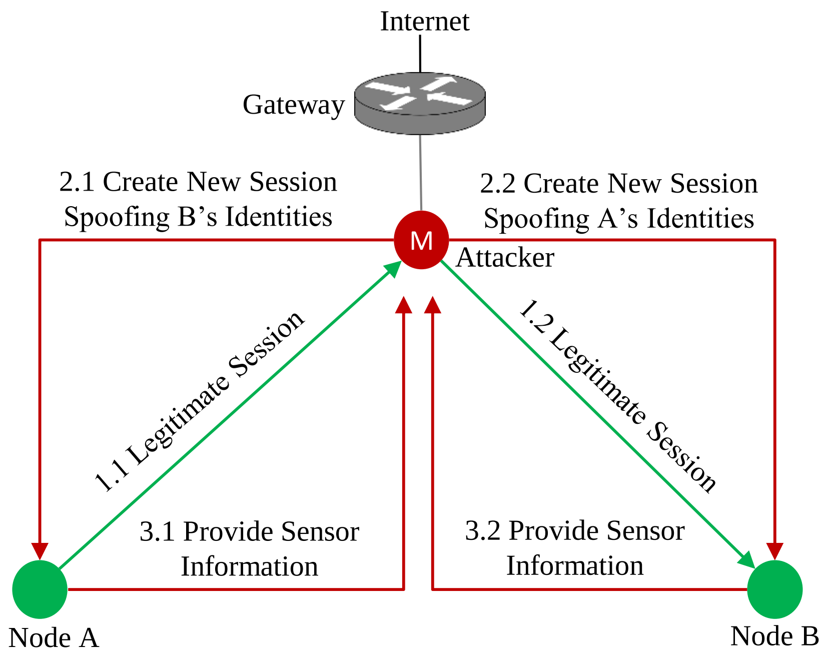

Internet
Gateway
2.1 Create New Session Spoofing B's Identities
2.2 Create New Session Spoofing A's Identities
M
Attacker
1.1 Legitimate Session
1.2 Legitimate Session
3.1 Provide Sensor Information
3.2 Provide Sensor Information
Node A
Node B

Figure 7. Man-in-the-Middle attack.

Modification: In this type of attack, a malicious node modifies messages that are routed through it. An adversary attempts to trick a victim node into performing unauthorized actions or revealing sensitive information. A scenario of the message modification attack is shown in Figure 8. A malicious Node M is located in the communication path between Node S and Node D. Node M modifies the content of a message sent by Node S before forwarding it to Node D. Node D may accept the altered message if it does not implement a security scheme that validates the integrity of a received message. As a result, the victim node may perform an action based on the modified information.

Fabrication: In this attack, a malicious node does not modify a message; instead, it inserts counterfeit information into the original message. The goal of the adversary is to create confusion between communicating peers and trick a victim node into issuing malicious commands. As shown in Figure 9, a malicious node inserts a forged header (k_{n+1}, v_{n+1}) into a request. The fabrication attacks threaten message integrity.

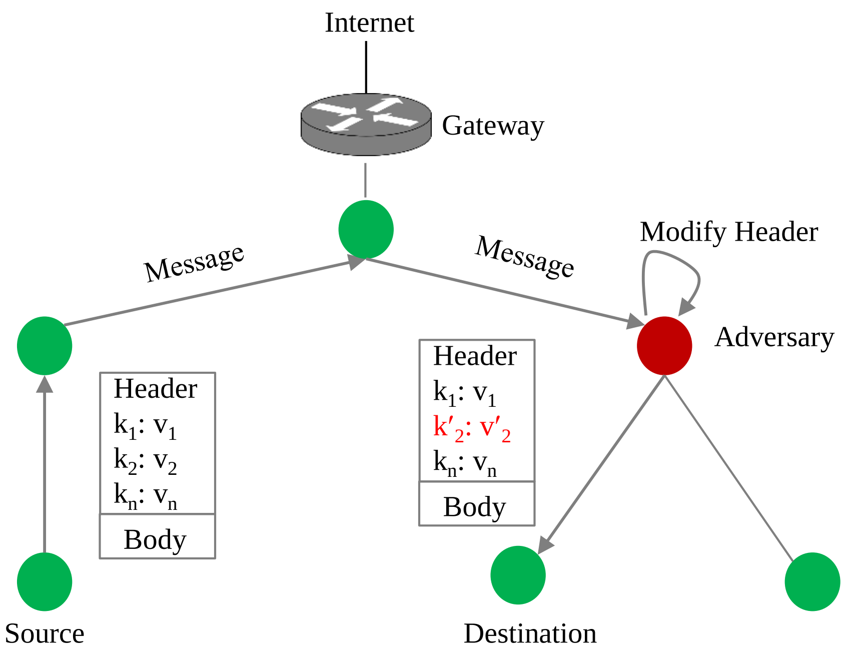


Figure 8. Message modification.

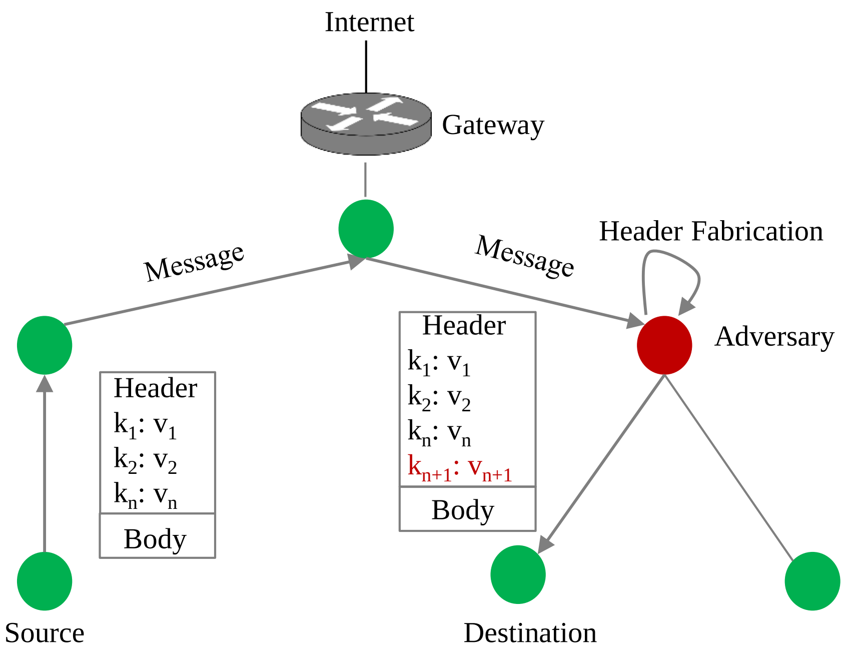


Figure 9. Message fabrication.

Message Replay: An adversary stores messages that are routed through, although the adversary is not authorized to do so. The adversary can also store a message if the victim node and adversary share the same radio signal. As shown in Figure 10, the adversary and source node share a radio signal although they are not located in the same subtree of an RPL network. The adversary eavesdrops on the communication link and stores the message exchanged between the source and destination nodes. Later, the adversary replays the message to impersonate the source node and get authenticated to the destination node. If an IoT node does not have a method to identify whether a message was delivered in the past, the node will consider a replayed message as a legitimate one and perform according to the instruction included in the message regardless of whether it is sent by an adversary.

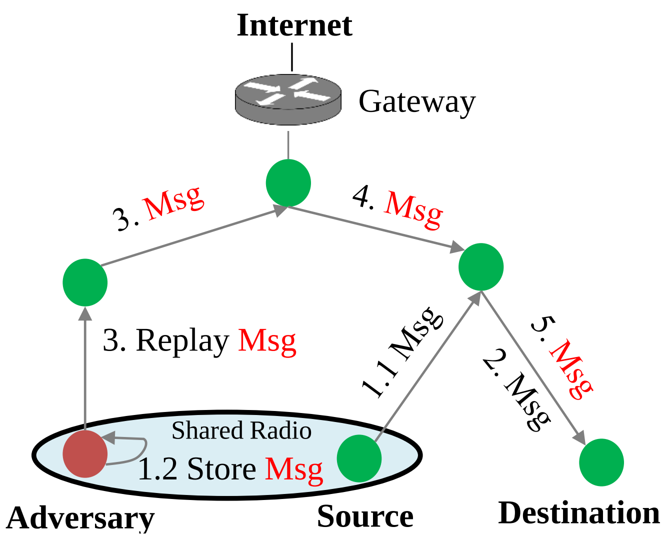


Figure 10. Replay attack.

4.4. Host-Based Attacks

User Compromise: Adversaries can adopt various social engineering techniques [98] that trick users into revealing their personal information and security credentials, such as name, date of birth, username, and password. Adversaries can send phishing emails and text messages to users to obtain usernames and passwords used for communicating with smart devices by impersonating a legitimate IoT service provider. As shown in Figure 11, an adversary attempts to obtain the credentials of the owner of a smart refrigerator by sending phishing text messages to the owner's smartphone.

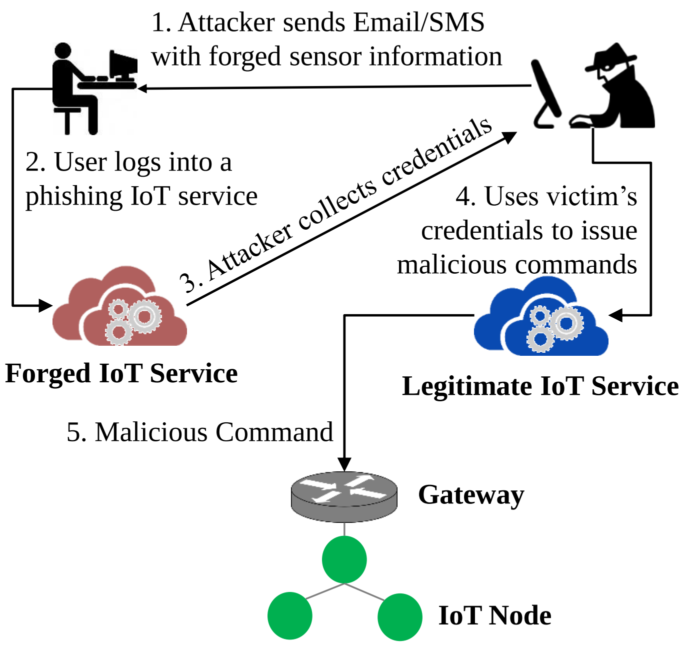

1. Attacker sends Email/SMS with forged sensor information
2. User logs into a phishing IoT service
3. Attacker collects credentials
4. Uses victim's credentials to issue malicious commands
Forged IoT Service
Legitimate IoT Service
5. Malicious Command
Gateway
IoT Node

Figure 11. User credential compromise.

Software Compromise: The vulnerabilities in web services, operating systems, and firmware can be exploited by adversaries to get access to IoT devices and networks. An attack scenario of the software compromise attack can be as follows. An adversary identifies that an IoT device runs a web server on port 80 by using port scanning tools [99] and techniques [100]. Next, the adversary uses password enumeration tools [101] to find the username and password used to log into the device. The web service may not have implemented a policy that locks the device after a certain number of login attempts or requires a complex password. As such, the adversary gets access to the device through password enumeration. After getting access to the device, the adversary can create a backdoor, such as a reverse shell, on the device to obtain sensor information. The adversary can also reconfigure the device to provide false sensor information, such as fake room temperature or health conditions, to its owner. As shown in Figure 12, in a connected vehicular scenario, an adversary reprogrammed a Roadside Unit (RSU) with malicious codes such that the RSU provided false information to smart cars.

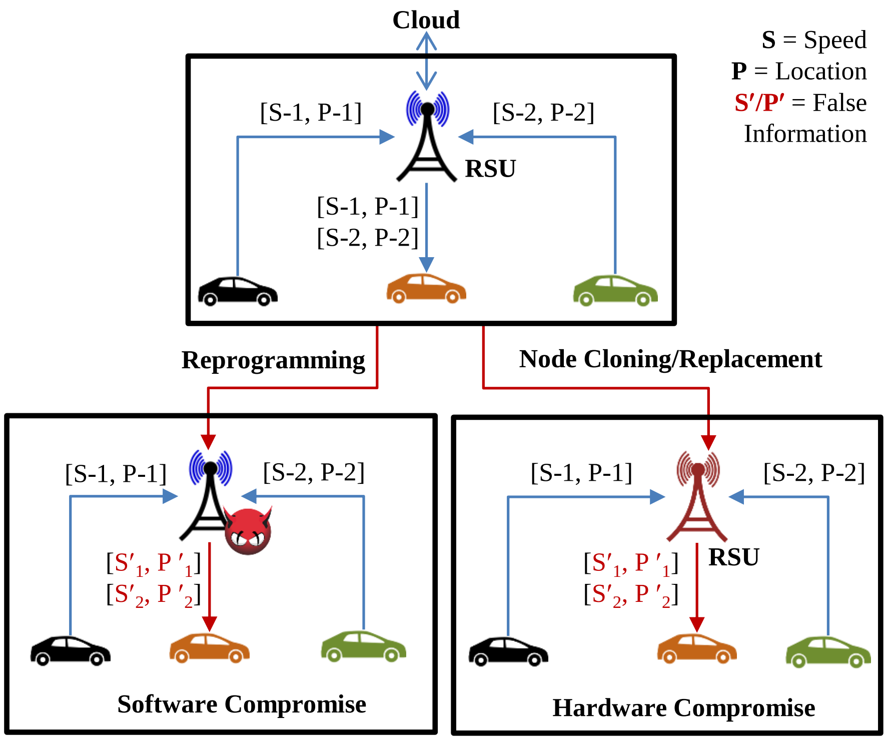


Figure 12. Software/hardware compromise threat model for Internet of Vehicles. RSU = Roadside Unit.

Hardware Compromise: The in-device storage of an IoT device contains sensor data, keying materials, and program code. An adversary can tamper with an IoT device to extract the embedded credentials and then use the credentials to impersonate a legitimate device. The adversary can obtain certificates and shared, public, and private keys stored on a device's memory by performing micro-probing and reverse engineering on the particular device. For example, RSUs are vulnerable to hardware compromise attacks, as they are left unattended after being installed on the side of the road (see Figure 12). RSUs and smart vehicles exchange traffic information, such as the distance between vehicles and road conditions. Adversaries can tamper with the RSUs to obtain their embedded credentials. Later, malicious RSUs can use the credentials to provide false traffic information to smart vehicles. False traffic information may lead to accidents.

4.5. Severity of IoT Attacks

We suggest three different types of severity levels to group the above mentioned IoT attacks. The severity levels are high, moderate, and low. Figure 13 shows the assignment of severity levels to IoT attacks.

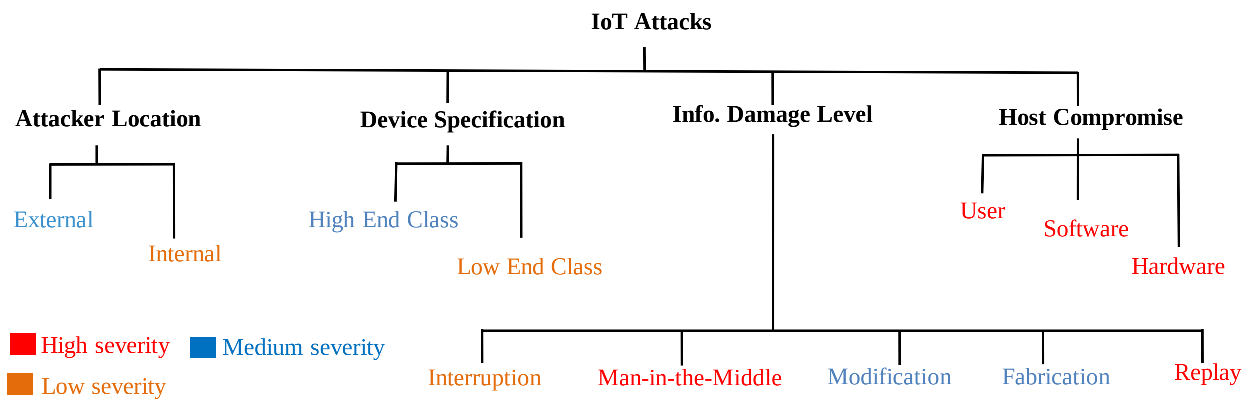


Figure 13. A taxonomy of IoT attacks.

The severity levels are assigned as follows:

1. **High-Severity Attack:** High-severity attacks can completely compromise an IoT system. Attacks of this category result in the loss of data confidentiality and integrity and unauthorized access to IoT networks and devices. The attacks that fall under this category are host compromise, man-in-the-middle, and replay attacks, because a successful attack allows an adversary to obtain credentials used for authentication and encryption as well as to perform actions without authentication.
2. **Moderate-Severity Attack:** An IoT system may be partially compromised by moderate-severity attacks. Attacks of this category have high impacts on the availability of services provided by smart devices. However, attackers may not have access to sensor information, devices, or networks. Attacks that may result in resource exhaustion can be considered moderate-severity attacks. Therefore, high-end, external, message modification, and fabrication attacks are included in this class.
3. **Low-Severity Attack:** In-device and in-transit information is not compromised by a low-severity attack. Moreover, a successful low-severity attack does not result in unauthorized access to networks and devices. Additionally, the availability of IoT services is not affected by low-severity attacks. As such, low-end, internal, and interruption attacks are included in this class.

4.6. Summary and Insight

In this section, we have examined various attack scenarios for IoT systems. We found that a successful high-severity attack can allow adversaries to take control of an entire IoT network. Therefore, penetration testing [102–104] should be performed periodically to assess the vulnerabilities in IoT systems and ensure the IoT devices, services, and networks are not compromised. Moreover, IoT service providers should take necessary steps to educate users and device owners about social engineering attacks so adversaries cannot compromise user credentials by sending phishing emails or text messages.

5. Security Requirements

The level of complexity of designing a security scheme varies from one smart system to another. The complexity level depends on the types of devices, connectivities, services, and applications present in an IoT system. Figure 14 presents a framework with the security complexity parameters. In the complexity framework, the changes of a parameter (such as application, device specification, and connectivity) in any dimension may increase or decrease the design complexity. For instance, design complexity increases with the increase in the number of communication interfaces in an IoT system. Let us consider connectivity complexity as an example. A smart device can use a non-IP network protocol for local network communications, such as things-to-things message exchange, and TCP/IP protocol for public network communications, such as things-to-cloud information exchange. The security scheme should be designed such that it can provide the same level of protection both for the IP and non-IP communications. Several other requirements and properties need to be included in the design of IoT security schemes. The details of these security properties are presented below.

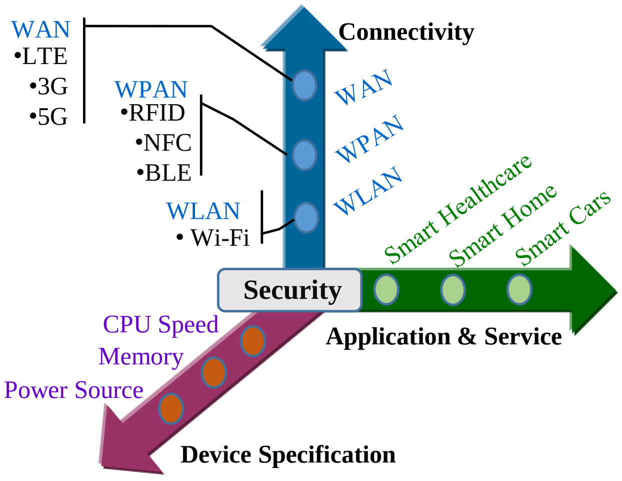


Figure 14. Complexity parameters for security solutions.

5.1. Access-Level Security Requirements

Authentication: This property enables a smart device to verify the identity of its communicating peer. Hence, a smart device ensures that devices or users with valid credentials get access to its services and resources. For instance, a Coordinator device authenticates a network administrator before allowing her to perform administrative tasks, such as remote reprogramming of a smart device, in an IoT network. Similarly, a user has to be authenticated to a smart device before the device performs actions according to a command issued by the users.

Access Control: The access control methods ensure that users and devices with valid credentials only get access to authorized services. This security property protects smart devices and services from unauthorized access. For instance, a patient's wearable medical sensors can be configured such that a physician can issue read and write commands to them. However, a nurse may only be authorized to issue read commands to the medical sensors.

Accounting: Accounting ensures that the activities of the entities of an IoT system can be used for auditing purposes. Security analysts use activity logs to investigate forensic incidents. An analyst can track the communications that took place between things and users, things and things, or things and clouds in the past as evidence to resolve disputes between users and IoT service providers or to find facts in cyber-criminal cases. As such, accounting services should have the ability to store logs and allow auditors to obtain logs for investigating cyber incidents.

5.2. Information Security Requirements

Integrity: This property allows an IoT device to verify the integrity of received messages. The integrity of in-transit and stored data can be compromised by modification and fabrication. Smart devices should be able to identify such altered requests and should not process them.

Information Protection: This property protects the privacy and confidentiality of the stored and exchanged information. Smart devices should avoid sending messages in plain text. These devices should adopt encryption techniques to preserve the confidentiality of the sensor information. Smart devices should implement policies that limit access to information and disclose to trusted parties to protect information privacy. As such, malicious entities will be prevented from learning security-critical information. For instance, there may be policies defined for an IoT network that do not allow a smart node to share sensor information with the neighboring nodes (devices that share the same radio signal).

Anonymity: This security property protects the privacy and confidentiality of the identity and locations of a smart device. Anonymity enables IoT devices to hide their identifiers and locations when they provide sensor information to users, clouds, and other devices. In a smart city, roadside sensors send traffic updates to cars, pedestrians, traffic lights, and clouds periodically. Anonymity ensures that the identity of the sensors and their deployment locations are not revealed when the sensors provide traffic information to smart cars and drivers.

Non-Repudiation: This property prevents parties that are involved in a communication, such as devices, gateways, clouds, and users, from denying their participation in exchanging messages. A medical sensor cannot deny sending a patient's health conditions that it has previously sent.

Message Freshness: This security property ensures that IoT service consumers receive the most recent sensor data. Smart devices are expected to provide real-time information on IoT environments. This property enables a service consumer to validate that received messages are most recent and not re-played. For instance, medical IoT services located in the cloud should be able to verify whether a medical sensor sends real-time updates on the patient's health conditions.

5.3. Functional Security Requirements

Interoperability: Heterogeneous hardware and software, such as radio transceivers and operating systems, are embedded with IoT devices. Security solutions should be designed and developed such that their deployment to an IoT system does not interrupt functional operations between heterogeneous things or prevent a device from communicating with its peer securely. For instance, a device with a newer version of a security method should be able to communicate with a device with the older version of the security scheme. Moreover, the same security scheme can be integrated with devices with different operating systems or software platforms. In this case, the security method should allow the devices to communicate to each other seamlessly without compromising message authenticity, integrity, and confidentiality. This property can be evaluated based on the amount of information (such as multiple versions of cryptographic algorithms, certificates, and keys) that a device needs to store in its memory to establish security associations (such as identify validation and session key establishment) with devices that have heterogeneous operating systems.

Scalability: According to recent research by Forbes [105], over the next five years, one million IoT devices will be commercialized per year. Therefore, security systems should scale well for IoT systems with a large number of smart nodes. A criterion for evaluating the scalability of a security method can be the architecture, centralized or distributed, on which the method is designed. For instance, distributed key management may be better than centralized key management for a smart city, which accommodates a large number of smart devices ranging from roadside sensors to smart cars.

Memory Efficiency: Smart devices have limited storage in RAM and ROM. Security methods should be designed such that cryptographic materials, such as keys and certificates, and program codes take minimal space in the flash memory (ROM). Security schemes should also be optimized to reduce memory (RAM) consumption during the execution of the cryptographic algorithms, such as identity verification, making authorization decisions, encryption, signature validation, and session key computations.

Minimal Communication Overhead: IoT networks are lossy and low powered and have limited network bandwidths. Therefore, security methods should minimize the total number of messages and bytes that devices are required to exchange between them to establish session keys. Moreover, radio-transceivers consume a significant amount of energy to receive and deliver network packets. The exchange of a minimal number of protocol messages will reduce energy consumption and increase battery life.

Minimal Computation Cost: IoT devices are embedded with low-power CPUs that operate at slower CPU clocks. Security schemes should avoid computation-intensive cryptographic operations in authentication and session key derivation to allow IoT devices to execute security algorithms faster. Moreover, the execution of such security schemes will enable devices to avoid burning too many CPU cycles, reduce energy consumption by CPUs, and ensure longer battery life.

Exception Handling Capability: This property allows IoT devices to provide a minimal level of service during unfavorable incidents, such as software glitches, malfunctioning hardware, dislocation environmental hazards, and denial-of-service attacks. This property also ensures that IoT devices can continue services, even in an anomalous situation, without compromising information security.

Resiliency: The security of an IoT system should not be compromised even if adversaries get unauthorized access to one or more nodes of the system. As such, this property allows security methods to avoid single points of failure on the IoT network. For instance, an adversary can compromise one or more nodes of an IoT system. However, security protocols should be designed such that the reaming nodes of the system can identify the compromised nodes and continue protecting the system from attacks by providing a set level of security.

5.4. Summary and Insight

In this section, we have identified the requirements for IoT security schemes. The access-level and information security properties must be included in the design of the security systems, as they ensure privacy, confidentiality, authenticity, and integrity of communications. In addition, an adequate amount of time should be spent on the design to meet the functional security requirements, because, unlike conventional digital devices, most of the IoT devices have limited resources, deal with real-time and security-sensitive information, and are required to be online 24/7.

6. Security Solutions

In this section, we review the existing security methods used for mitigating IoT attacks. We classify the contemporary security schemes into three domains: end-device security, communication security, and service security. Figure 15 presents the classification of the security solutions from these three domains. We summarize prior solutions protecting the IoT devices from different attack surfaces and mitigating vulnerabilities in Table 6.

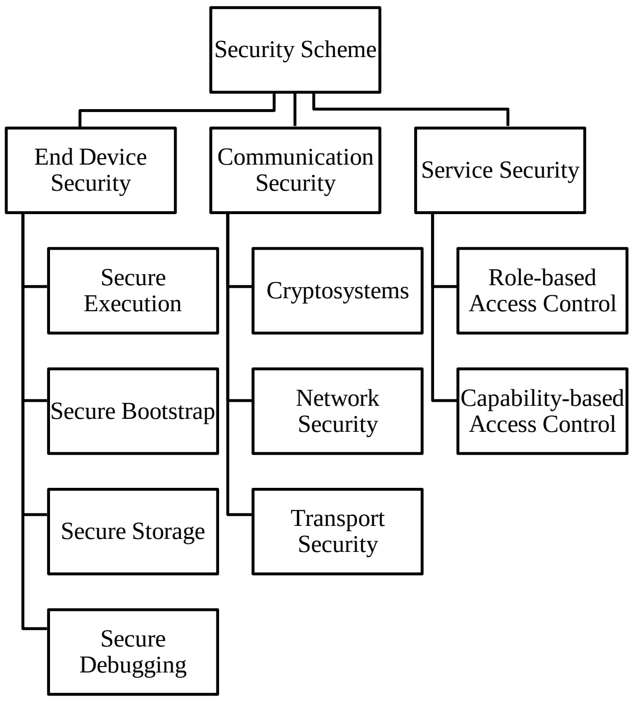


Figure 15. Classification of the existing security schemes.

Table 6. Summary of the available solutions to mitigate IoT threats.

Aspects	Threats	Mitigation	Schemes
Device Security	Software Compromise	Secure Execution	[106–109]
		Secure Bootstrap	[110]
	Hardware Compromise	Secure Debug	[111–113]
		Secure Storage	[114–116]
Network Layer Security	Identity Impersonation	Authentication and Encryption Using Host Identity Protocol (HIP)	Base Exchange [117]
			Distributed [118]
			Tiny Exchange [119]
	Information Disclosure		Diet Exchange [120,121]
			Slimfit [120]
			Pre-Shared Key [122–125]
			Lightweight [126,127]
Transport layer Security	Identity Impersonation	Authentication and Encryption Using DTLS	Certificate-based [128]
			DTLS-PSK[122,125,129]
	Information Disclosure		Modified [130]
			Delegated [131–133]
Application Layer Security	Unauthorized Access	Role-based Access Control (RBAC)	Centralized [134]
			Context-aware [135]
		Capability-based Access Control (CapBAC)	Centralized [136–138]
			Distributed [139–141]
			Context-Aware [142]
			OAuth Compliant [143,144]

End-device security protects IoT devices from host compromise attacks, such as software and hardware tampering. End-device security methods ensure the integrity of the embedded software and system booting process, protect on-chip storage from micro-probing, and provide a secure execution environment that isolates the computation of the trusted and untrusted software. Communication security provides authentication for device-to-device communications and ensures confidentiality, integrity, and non-repudiation of the information exchanged between devices. Finally, service security protects IoT devices and their resources from unauthorized access.

Since a comparative survey is one of the key objectives of this article, the performance of the aforementioned security solutions is evaluated in terms of the functional security properties described in Section 5.3. The analysis eventually determines the usability of the existing security methods for IoT systems. Table 7 summarizes the performance metrics of interest in this regard.

Table 7. A list of metrics used to analyze the performance of contemporary security schemes.

Property	Highlights
Memory Requirement (MR)	The total number of bytes required to store public keys (m_{pk}), private keys (m_{sk}), and certificates (m_{crt}) in volatile and non-volatile memory. $MR = \sum(m_{pk}, m_{sk}, m_{crt})_{ROM} + \sum(m_{pk}, m_{sk}, m_{crt})_{RAM}$

Table 7. *Cont.*

Property	Highlights
Communication Overhead (CO)	The total number of messages (m_t) and bytes (b_t) exchanged until a session key is negotiated. In addition, the amount of energy (e_{co}) consumed by a radio transceiver for exchanging messages. $CO = \sum(m_t, b_t, e_{co})$
Computation Complexity (CC)	The number of arithmetic operations (o_t), such as addition, subtraction, multiplication, division, and modular exponentiation, required to compute a session key. In addition, the amount of energy (e_{cc}) consumed by the CPU for performing cryptographic computations. $CC = \sum(o_t, e_{cc})$
Resilience	The ability to provide services under Denial-of-Service attacks
Scalability	The ability to accommodate a large number of IoT devices
Interoperability	The ability to negotiate a cipher suite to establish a secure association with heterogeneous devices

6.1. End Device Security

6.1.1. Secure Execution Environment

Software attacks compromise the software executed on the IoT devices. A Secure Execution Environment (SEE) can provide essential security for IoT software in the face of untrusted applications and operating systems. A SEE can be achieved by using trusted computing techniques, such as Trusted Platform Module (TPM) [106].

A TPM chip ensures protected execution of security-sensitive applications and operating systems. A processing unit embedded with the TPM chip isolates the execution of the trusted software and untrusted software. The isolation is performed based on either physical or logical separation of processes. In physical separation, trusted software is executed on a dedicated secure processor, such as the secure co-processor developed by IBM [107]. In contrast, in logic separation, a single processor is capable of supporting a secure mode, which isolates sensitive and untrusted code by using an additional software layer [145] or hardware support, such as Intel TXT and SGE platforms [108] and ARM TrustZone [109].

6.1.2. Secure Bootstrap

A secure boot routine [110] can prevent malicious applications from loading during the system start-up process. A secure boot routine is a piece of code that can detect any modification in the operating system and system software during the start-up phase. Thus, a secure boot routine can ensure system integrity and bring the system to a known and trusted state. The bootstrap code is stored in the flash memory, such as NOR or NAND flash, and is executed directly from there. This ROM-based execution approach prevents an adversary from perceiving the bootstrap process. The complete bootstrap solution consists of additional security features, including a software update routine which receives and integrates new code and libraries to upgrade the system to a new version of the software image.

6.1.3. Secure Storage

IoT devices store cryptographic keys, sensor readings, and system images in their memories. The secure storage ensures protected access to this security-sensitive data. The on-chip memory can be designed using one-time programmable technology, such as read-only memory and poly-silicon fuses [114–116], to store codes and credentials securely. As a result, reprogramming attacks cannot modify or replace information stored in the secure storage.

6.1.4. Secure Debug Interface

The debugging interface is primarily used during development and manufacturing of a device. This interface allows debugging of on-chip applications and loading system images into flash memory. It also enables system administrators or device manufacturers to find and fix errors that occur during the lifetime of a system. The JTAG [146] is one of the most widely used debugging interfaces for chips. This interface can be a potential attack surface for adversaries. Adversaries can exploit the JTAG interface to read on-chip registers and memories and reprogram the system image (overwrite the trusted system image with the malicious one). Secure JTAG implementation has been proposed in [111–113] to defend against such malicious activities.

6.2. Communication Security

IoT devices use CoAP (Constrained Application Protocol) [71] as an application layer protocol. The CoAP implements IP-based communication protocols to achieve pervasive interactions between connected devices. The communication protocols implement various security schemes to ensure authenticity, confidentiality, and integrity of the information. Research on IoT security proposes new standards and methods for IoT systems. These security solutions introduce additional layers or modify existing layers in the IoT protocol stacks, as shown in Figure 16. In this section, we first review the cryptographic algorithms that are suitable for IoT devices. Next, we survey security methods that provide network and transport layer security for IoT systems.

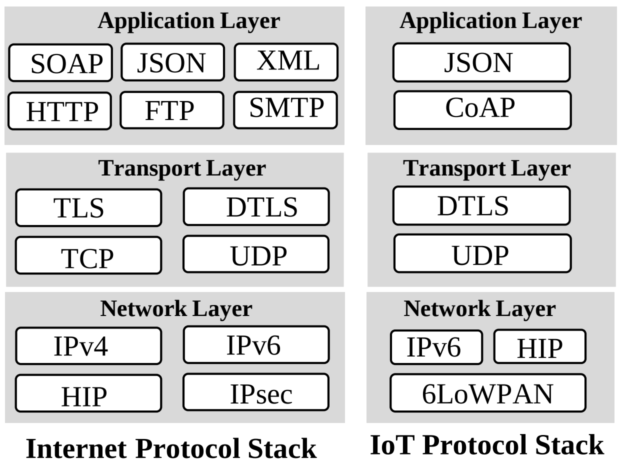


Figure 16. Internet stack vs. IoT stack [147].

6.2.1. Cryptosystems

Cryptography: Security solutions that ensure message integrity, confidentiality, and non-repudiation as well as provide authentication and authorization are developed based on symmetric and public key cryptography. Each of these schemes has some advantages and limitations [148]. The symmetric key cryptography (SKC)-based schemes are not memory efficient because they consume a considerable amount of memory to store keying materials. These schemes do not scale well because of the complicated key distribution mechanisms. On the other hand, the pubic key cryptography (PKC)-based schemes are not energy efficient because they have noticeable communication and computation overheads. Moreover, the PKC-based schemes take considerable time to verify security credentials, such as certificates, and establish session keys.

Wang et al. [149] presented an experimental study that shows PKC-based schemes are easy to implement compared to SKC-based schemes. The study also finds that, unlike SKC-based schemes, PKC-based schemes do not require a sophisticated key management mechanism, such as pair-wise key sharing and key pre-distribution.

Although PKC-based protocols are found advantageous, these security schemes cannot be utilized by the resource-constrained IoT systems as-is. Areas that require further study are: *(i)* resource-efficient implementations of PKC-based algorithms; *(ii)* certificate and key distribution mechanisms with minimal communication overheads; *(iii)* an in-depth security analysis of the resource-efficient designs of PKC-based protocols to confirm that the PKC-based schemes can at least provide the same level of security as the SKC-based schemes do.

Analysis of Cryptographic Algorithms: The most popular cryptographic protocol used for encryption, decryption, and signature in the PKC is RSA, which is designed based on Integer Factorization (IF). RSA-based security protocols use the Diffie–Hellman (DH) key exchange algorithm to establish shared keys. The DH algorithms are designed based on Discrete Logarithms (DL). Both of the IF and DL algorithms require communicating parties to perform modular exponentiation operations that are resource intensive in terms of CPU cycles and power consumption. According to the National Institute of Standards and Technology (NIST), communication protocols should use RSA keys of length 2048 bits to achieve a good degree of security [150]. However, RSA with key lengths of 2048 bits consumes a significant amount of energy because of the modular exponentiation operations of IF and DL algorithms [151], such as public key = $\{a^b(mod\ n) \mid a, b\ prime\ number\}$. Therefore, the RSA is not suitable for resource-limited IoT devices. SecFleck [152] achieves faster RSA operations through hardware support. However, the hardware-based RSA implementation increases the size of the chip, as it requires a large silicon area [153]; therefore, such implementation of RSA protocols may not be suitable for mobile and wearable IoT devices.

An alternative to RSA cryptography can be cipher protocols based on Elliptic Curve Cryptography (ECC) and Advance Encryption System (AES). The ECC is an asymmetric algorithm that can provide equal safety by the use of shorter key length. An ECC key of size 248 bits can provide the same degree of security as an RSA 2048-bit key [150]. As such, ECC-based security protocols, such as Elliptic Curve Diffie–Hellman (ECDH) used for session key computation and Elliptic Integrate Encryption used for protecting message confidentiality, are much more memory efficient than the RSA algorithm. Moreover, unlike the DL-based security methods, the arithmetic operations of ECC-based security schemes do not require computations of modular exponentiations. The ECC-based schemes only include addition and multiplication operations. Therefore, ECC-based algorithms have lower energy consumption than IF- and DL-based algorithms and are suitable for resource-constrained smart devices.

Previous studies [154–156] analyzed the feasibility of using ECC-based solutions by resource constrained devices without compromising the security requirements that we identified in Section 5. These studies demonstrated the applicability of ECC-based cryptosystems to IoT devices through experimental evaluations. On the other hand, security protocols described in [157] advocate the use of symmetric encryption algorithms, such as AES for IoT systems. However, the adoption of ECC and AES has been slowed because there still exist significant differences between the requirements of security processing and the capabilities of IoT devices [158].

A performance comparison of the above-mentioned cryptographic algorithms is presented in [159]. The authors evaluate the performance in terms of memory requirements, energy cost for cryptographic computations, and communication overheads for establishing a session key. The analysis shows that the energy cost of public-key cryptography is minimal, if not negligible, for applications that require infrequent authentication and key exchanges. However, which cryptographic algorithm to be used depends upon the security requirements of the respective IoT applications since applicability of the cryptographic algorithms varies according to the capabilities, configurations, and operations of IoT devices.

We present a list of cryptographic algorithms that can be suitable for various IoT components in Table 8. As shown in Figure 17, smart devices and sensors are the most resource-constrained entities in an IoT system. As such, these devices should avoid using the cryptographic schemes that include modular exponentiation operations and require to have longer keys, such as IF and DL, and adopt ECC-based cryptologic schemes.

Table 8. Recommendations on cryptographic primitives to be used at different IoT components [160].

Property	**Constrained/Intranet**			**Public/Internet**	
	Things-to-Things	**Things-to-Coord.**	**Coord.-to-Gateway**	**Gateway-to-Cloud**	**Cloud-to-Controller/User**
Data Size	<1 KB	<1 MB	<512 MB	<1GB	<1 GB
Cryptography	ECC-160/224	ECC-224/256	ECC-256/384 RSA-2048/3072	ECC-384/512 RSA-2048/3072/7680	ECC-384/512 RSA-2048/3072/7680
Key Exchange	ECDH	ECDH	ECDH/DH	ECDH/DH	ECDH/DH
Enc/Dec	AES-112/128	AES-128/192	AES-128/192	AES-192/256	AES-192/256
Hash	MD5/SHA-1	SHA-1	SHA-2/SHA-3	SHA-2/SHA-3	SHA-2/SHA-3
Signature	ECDSA	ECDSA	ECDSA/DSA/RSA	ECDSA/DSA/RSA	ECDSA/DSA/RSA

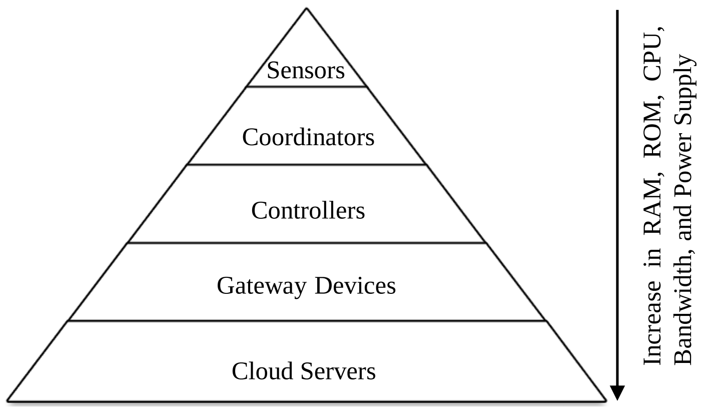
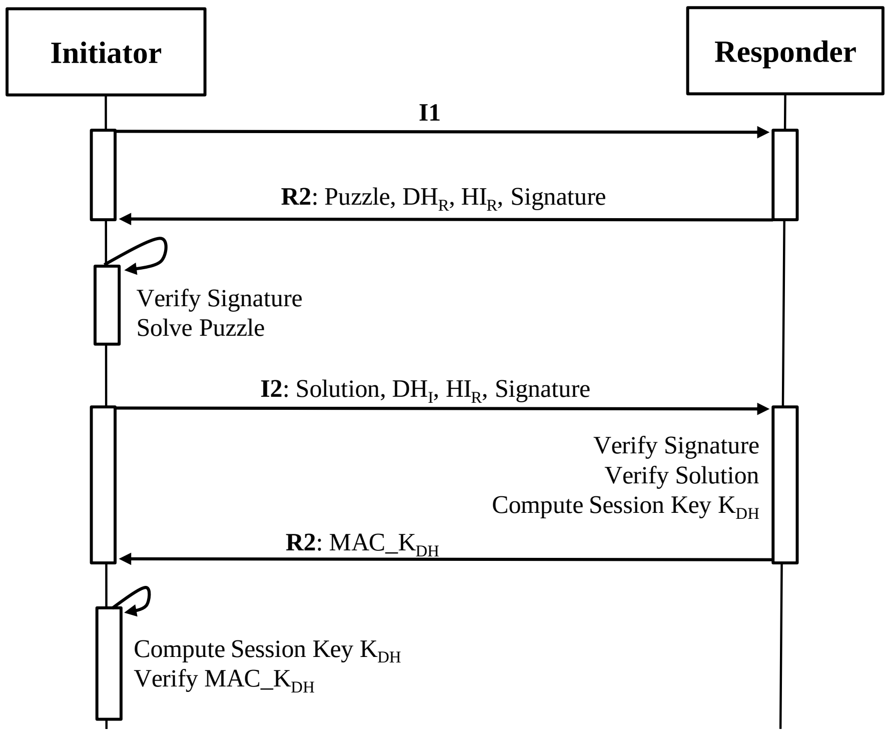


Figure 17. Device hierarchy.

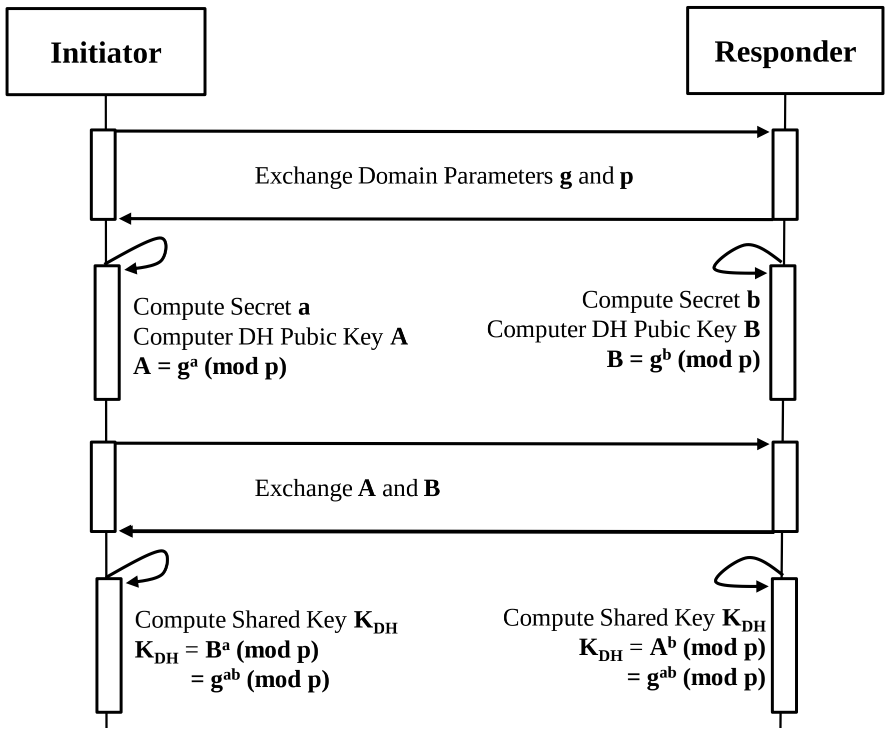
6.2.2. End-to-End Network Security

Considering the functional security requirements, such as global identification, mobility, and multiplicity of smart objects, the Host Identity Protocol (HIP) [117] can be a suitable solution for mutual authentication and key establishment between two communicating peers. The HIP introduces an additional layer (Host Identity layer) between the transport layer and the IP layer to resolve the problems caused by the dual role of an IP address: identifier and locator. However, the HIP cannot be applied directly to the resource-constrained IoT devices and networks because the authentication process of HIP requires communicating devices to perform computation-intensive cryptographic operations. Various lightweight versions of the conventional HIP are proposed to make HIP suitable for IoT systems. In the following sections, we provide a survey on these HIP-based security schemes.

HIP-Based Exchange (HIP-BEX): HIP-BEX [117] enables communicating devices to perform mutual authentication and establish session keys. We present the authentication and key establishment process in Figure 18. In HIP-BEX, communicating devices identify and authenticate themselves by using cryptographically protected host identifiers (HIs) and the RSA cryptography. Four types of messages are exchanged between an Initiator (a sender) and a Responder (a receiver). The Initiator starts the communication by sending an I1 message. The Responder replies with an R1 message, which contains a puzzle, the Responder's DH public value (DH_R) and host identifier (HI_R), and an RSA signature. The Initiator verifies the signature and authenticates the Responder. The Initiator also solves the puzzle and computes a DH session key (K_{DH}) using the DH public value DH_R. The Initiator sends an I2 message that consists of the solution, the Initiator's DH public value (DH_I) and host identifier (HI_I), and a signature. The Responder validates the signature and authenticates the Initiator. Next, the Responder computes the K_{DH} using the Initiator's DH public value DH_I. Finally, the Responder replies with an R2 message that includes a signed MAC calculated with the K_{DH} confirmation. Hence, a shared secret is established between HIP peers.



(**a**) Interaction flow of HIP-BEX.



(**b**) Diffie–Hellman key exchange

Figure 18. HIP-Based Exchange (HIP-BEX).

In HIP-BEX, both the Initiator and the Responder have to perform resource-intensive cryptographic operations involved in the computation of DH public and session keys. As shown in Figure 18b, modular exponentiation operations, such as $g^a\ (mod\ p)$, $g^b\ (mod\ p)$, and $g^{ab}\ (mod\ p)$, have the highest computation overhead on the IoT device. Additionally, there are computation costs for signature generation and verification that cannot be ignored for highly resource-limited devices.

Distributed HIP (D-HIP): D-HIP [118] proposes a collaborative scheme to reduce the computation cost for modular exponentiations. D-HIP delegates the DH operations to resource-rich proxy nodes co-located with the Initiator and Responder nodes in an IoT network. The Initiator selects a set of proxies and delegates the computation load for key exchange to them. After the exchange of the I1 and R1 messages, the Initiator splits its secret exponent a into multiple blocks $a_1, a_2, ..., a_n$ such that $\sum_1^n a_i = a$. The Initiator sends the blocks to the proxies. Each proxy receives a unique block a_i, computes its part of the Initiator's public DH key $A_i = g^{a_i}\ (mod\ p)$, and sends it to the Responder. The Responder receives all the parts from the proxies and computes the Initiator's public DH key (A) as:

$$A = \prod_1^n A_i = g^{\sum_1^n a_i}\ (mod\ p) = g^a\ (mod\ p)$$

Afterward, the Responder sends its secret exponent b to the proxies. Each proxy computes its part of the DH key $g^{a_i b}\ (mod\ p)$ and sends to the Initiator. The Initiator computes the session key K_{DH} as:

$$\prod_1^n g^{a_i b}\ (mod\ p) = g^{\sum_1^n a_i b}\ (mod\ p) = g^{b \sum_1^n a_i}\ (mod\ p) = g^{ab}\ (mod\ p)$$

Although the collaborative scheme for key establishment reduces computation overhead, it has the following disadvantages:

- The time to set up a session key increases significantly because the Initiator and Responder have to exchange a considerable number of messages with the proxy nodes.
- The proposed scheme assumes that the Initiator is resource constrained. Therefore, the proxy nodes compute the Initiator's DH public key and session key. However, the Responder can also be resource constrained and can delegate DH public and session key computation tasks to the proxy nodes. As a result, the collaborative scheme will contribute more to the communication overheads for exchanging protocol messages and will increase the key establishment time.
- If a single proxy fails to compute its part of the DH key correctly the D-HIP returns to the states where it selects proxy nodes and distributes blocks of its secret key. A malicious proxy can exploit this property to perform DoS attacks. The malicious proxy can avoid DH key computation and provide a false DH key to the Initiator. Hence, the malicious proxy can force the Initiator to perform proxy node selection and key distribution repeatedly.

HIP Tiny Exchange (HIP-TEX): HIP-TEX [119] proposes a distributed key exchange scheme to reduce the cryptographic overheads for HIP-BEX. HIP-TEX replaces the DH key agreement with a method that encrypts a session key using the public key cryptography. Additionally, HIP-TEX enables IoT devices to offload cryptographic computations to a set of proxy nodes in a collaborative scheme. The proxy nodes are resource-rich devices compared to IoT devices. HIP-TEX has less computation cost compared to HIP-BEX and D-HIP due to the new design decisions. However, HIP-TEX has similar disadvantages as D-HIP because both of the methods are based on a distributed collaboration scheme. As such, the HIP-TEX increases communication overhead and secret key setup time and is vulnerable to DoS attacks.

HIP-Diet Exchange (HIP-DEX): HIP-DEX [121] utilizes ECC to reduce the computation costs involved in HIP-BEX. HIP-DEX uses a long-term Elliptic Curve Diffie Hellman

(ECDH) public value as a Host Identifier. Thus, HIP-DEX avoids the cost of computing an ephemeral DH public key, unlike HIP-BEX. HIP-DEX also eliminates the computation cost of the modular exponentiation operations involved in the DH key agreement by adopting the ECDH key establishment method; therefore, HIP-DEX can be suitable for resource-limited devices. Moreover, HIP-DEX can be applicable to lossy networks, as it provides an aggressive retransmission scheme to cope with a higher packet loss. However, the ECDH-based key exchange can still be too heavy to be supported by a highly resource-constrained IoT device, such as an IoT device with 8 MHz CPU [24].

Compressed DEX (HIP-Slimfit): In [120], a packet compression layer (Slimfit) is proposed between the network and HIP-DEX layer to reduce communication overheads for exchanging HIP-DEX messages. The Slimfit layer compresses the outgoing HIP-DEX packets and sends them to the network layer. On the other hand, the Slimfit layer receives the incoming packets from the network layers, decompresses the packets, and forwards them to the HIP-DEX layer. Hence, the Slimfit layer reduces communication overheads for packet fragmentation, reassembly, and retransmission in the lossy networks. The packet processing time at the layers, such as the IPSec layer, located below the Slimfit layer, is also minimized as the Slimfit layer reduces the size of the packets.

HIP Pre-Shared Key (HIP-PSK): The HIP-PSK [122] is a variant of HIP-DEX. The HIP-PSK does not use the ECDH key agreement; instead, it relies on a pre-shared key-based key exchange scheme. In the HIP-PSK, a trusted entity named Domain Manager (DM) allows legitimate devices to join an IoT network. Every IoT device of a smart system shares a secret key (PSK) with the DM. The DM authenticates a joining device based on its shared key and then provides the device with network-access credentials (NACs), such as such as Layer-2 keys and polynomial shares [123]. After successful authentication, the DM and device compute a session key using the CMAC [124] as $K_s = CMAC(PSK|puzzle|solution)$. The DM uses the session key K_s to send the NACs to the device securely. After joining the network, devices use Host Identity Tags and NACs for authentication and session key establishment. Two communication devices use the AMIKEY key agreement [125] scheme to compute a session to encrypt messages.

The HIP-PSK is lightweight compared to HIP-BEX and HIP-DEX because the computation-intensive cryptographic primitives, such as DH and ECDH key exchange, are removed. However, an adversary can allow malicious devices to join a network by compromising the DM. As such, the HIP-PSK is vulnerable to single-point-of-failure. Furthermore, an adversary can tamper with an IoT device to extract the pre-shared key to join a network and perform identity impersonation attacks.

Lightweight HIP (LHIP): LHIP [126] does not implement any of the security mechanisms used by HIP-BEX and HIP-DEX. LHIP does not perform host authentication and message encryption to obtain simplicity and avoid cryptographic overheads. Although messages are exchanged in plain text between peers, the LHIP provides a minimal degree of security that uses hash chains to authenticate succeeding messages and can detect session hijacking.

Analysis and Comparison: We analyze and compare the outcomes of HIP-based methods listed in Table 9 using the metrics defined in Table 7. It can be noted that we have excluded the HIP-BEX protocol in the comparative analysis because of its inherent inaptness at constrained devices: HIP-BEX is the most computationally expensive method among the available HIP-based schemes. We notice that the collaborative HIP schemes [118,119] are potentially advantageous for IoT systems because of their low requirements of computation and memory. The collaborative schemes, however, raise congestion in the network due to additional communications among the collaborators themselves and between collaborators and IoT devices. Thus, these schemes eventually experience high communication overhead.

Table 9. A comparison between HIP-based schemes. Notations used for the security properties are: I = Interoperability, S = Scalability, R = Resiliency, CO = Communication Overhead, MR = Memory Requirement, and CC = Computation Complexity. A security property can have three difference values: high (★★★), medium (★★), and low (★). A value is assigned to a security property based on a HIP scheme's performance to support that property.

Scheme	Approach	Key Exchange	I	R	S	CO	MR	CC
HIP-DEX [121]	Standalone	Elliptic Curve Diffie Hellman	★★★	★★★	★★★	★★	★★	Maxium
Slimfit [120]	Standalone	Elliptic Curve Diffie Hellman	★	★★★	★	★★★	★★	↑
HIP-PSK [122]	Standalone	Pre-Shared key	★	★	★	★	★	
D-HIP [118]	Collaborative	Diffie Hellman	★★	★★	★★	★	★★	
HIP-TEX [119]	Collaborative	Public Key	★★	★★	★★	★	★★	
LHIP [126]	Standalone	Not Available	★★★	★★★	★★★	★★★	★★★	Minimum

The analysis reveals that HIP-DEX [121] can be adopted by IoT systems that are not computation limited, e.g., an IoT system which consists of relatively resource-rich smart nodes. Both the HIP-PSK [122] and Slimfit [120] are appropriate for IoT applications because they are efficient in terms of memory, computation, and communication requirements. Neither of them, however, offer any reasonable benefits regarding scalability and interoperability. In the HIP-PSK scheme, each IoT device needs to share a secret key with the Domain Manager. With a network of a large number of IoT devices, the distribution of the secret keys, thus, becomes a challenging task. Similarly, in Slimfit, it cannot be ensured that all the devices of an IoT network have the Slimfit layer. Therefore, the interoperability between devices with and without the Slimfit layer cannot be guaranteed. Furthermore, it is a complicated task to embed the Slimfit layer with a large number of IoT nodes. Finally, although the LHIP [126] protocol is found resource efficient, this scheme does not incorporate some of the essential security requirements, such as authentication and confidentiality.

6.2.3. End-to-End Transport Security

The application layer protocol CoAP [71] is primarily designed for resource-constrained devices. As such, most IoT devices adopt CoAP instead of HTTP for device-to-device communications [161]. CoAP runs on the connectionless UDP to achieve multicast supports, such as group communications, as well as to avoid packet fragmentation, loss, and retransmission, and message delivery delays in lossy networks. CoAP relies on DTLS [35] to achieve authentication, key management, and connection protection. As shown in Table 10, various authentication protocols, such as pre-shared key, raw public key, and digital certificates, are supported by the DTLS. The conventional DTLS schemes require numerous message exchanges to set up a secure connection. As such, IoT devices that operate in a lossy network cannot adopt the DTLS schemes as-is. Various lightweight versions of the DTLS are proposed for the constrained devices and networks, which reduce communication and computation overheads from IoT nodes. In this section, first, we survey the DTLS-based security methods designed for IoT systems. Next, we provide an analytical comparison between these security systems.

Certificate-based DTLS: Previous research [128] proposed a certificate-based DTLS scheme for mutual authentication. The proposed scheme uses X.509 certificate and relies on the RSA cipher suite for communication security. To achieve mutual authentication, communicating peers first exchange their certificates and then validate the authenticity of the certificates by using the RSA signature verification method. The proposed scheme ensures tamper-proof generation and storage of keying materials, such as RSA keys and X.509 certificates, with the help of a TPM chip [106] embedded with an IoT device. The TPM chip also provides hardware support for faster RSA operation. The authors implemented the proposed solution on Opal sensor nodes with 48 MHz processors and 48 KB RAM [25] and provided a performance evaluation regarding computation, communication, and memory overheads for the DTLS handshaking and data transmission phases. Although the proposed solution showed performance improvement for Opal sensor nodes, the scheme may not perform well for a heavy resource-constrained device, such as T-Mote Sky [24] and Z1-Mote [23], because the storage capacity and computation speeds of these devices are less than the Opal sensor nodes. Additionally, the proposed system cannot process certificate chains and revocation lists. Finally, the proposed approach is not cost effective for an IoT system that comprises a significant number of nodes because there is a considerable cost of buying the TPM chips and embedding them with every device of the smart system.

Table 10. CoAP(s) security modes and their properties.

DTLS	Security Mode	Property
Disabled	No security	– No protocol security – Relies on underlying layers security
Enabled	Pre-shared key	– List of keys are generated – Each key includes a list of nodes to be communicated with
	Raw public key	– Asymmetric key pair – A public key is trusted but not associated with a certificate
	Certificate-based public key	– Asymmetric key pair – X.509 certificate binds to a public key – Signed certificate

DTLS Pre-Shared Key (DTLS-PSK): The authors in [122] proposed a pre-shared key-based authentication method [129] for DTLS that allows IoT devices to join a network securely. In the proposed system, every device of an IoT network shares a secret with a trusted entity of the network named as Domain Manager (DM). An IoT device and the DM perform the DTLS-PSK procedure for authentication. The DM provides network access credentials to the IoT device after successful authentication. However, once a device joins the network, the proposed system does not follow the DTLS protocol to protect communication between two peers. Instead, devices use the AMIKEY key agreement [125] to ensure the confidentiality of messages. The DTLS-PSK scheme has less communication overhead compared to the public key-based DTLS scheme because it reduces the number of exchanged messages. However, the scheme is vulnerable to a single point of failure because an adversary can take control of an entire IoT system by compromising the DM.

Modified DTLS: The MTU of an IoT network is 127 bytes [68]. The DTLS headers are too large to fit in a single MTU and sent in multiple fragments. To reduce the size of the DTLS headers, a header compression technique is proposed in [130]. In the proposed solution, an IoT Gateway performs the header compression upon observing DTLS handshakes. The compression reduces the size of the DTLS record and handshake headers and the number of packet fragments. The improvement in the packet size reduces the number of fragments, packet processing time, and message delivery time. The energy consumption is also reduced because a sender and receiver have to process fewer packets. Although the header compression scheme increases resource efficiency, its design does not consider the function for providing backward compatibility with standard DTLS protocols. A device without the header compression module in the DTLS layer cannot communicate with a device that has header compression capabilities.

Delegation-based DTLS: The authors in [131] separate the DTLS protocol into two phases (handshake and encryption), to reduce message delivery overheads, such as packet fragmentation, reassembly, loss, and retransmission. In the proposed scheme, an IoT device delegates the handshake phase to an IoT Gateway. The Gateway performs the handshake for the IoT device and sends the session key to the device. For rest of the communication, the device only performs encryption or decryption with the session keys. In theory, the proposed method should reduce communication overhead for the constrained IoT devices. However, the research work does not provide any proof-of-concept implementation and performance analysis. Moreover, an adversary can learn the communications if it compromises the Gateway to get access to the session key.

The research in [132] provides a solution that enables IoT devices to process certificate chains and revocation lists. In the proposed scheme, an IoT device delegates the certificate validation task to a Delegation Server (DS), a rich-resource device co-located with IoT nodes in the same IoT network. The DS checks the validity of the certificates, ensures that certificates are not listed in the revocation list, and verifies the certificate chain if necessary. The proposed solution also supports the session resumption [133] feature of the DTLS protocol. Security contexts of a connection are preserved in the DS, which allows resuming the connection after it is terminated. In the session resumption, the DS sends the stored security contexts to the constrained device after certificate validation.

Although both of the schemes mentioned above increase resource efficiency of the DTLS protocol, they are vulnerable to a single point of failure because they have to trust the Gateway and DS. An adversary can perform resource exhaustion–type attacks on trusted devices to prevent them from participating in the handshake phase. As a result, IoT devices cannot establish a secure connection with their peers and provide services.

Analysis and Comparison: We analyze and compare the outcomes of DTLS-based methods in terms of the metrics listed in Table 7 to determine the extent to which we can apply these existing security schemes on the IoT systems. The findings are briefly summarized in Table 11. Because of their low memory, computing, and communication overheads, the delegation-based DTLS methods [131,132] are advantageous for an IoT environment. However, these schemes severely suffer from a single point of failure and DoS attacks. These solutions are also not appropriate from a scalability perspective, since the delegation server is required to manage a significant number of requests resulting from the exponential increase in the deployment of smart devices. Conversely, the certificate-based DTLS solutions [128] are beneficial in terms of interoperability, scalability, and resilience. Nevertheless, because of their resource-hungry characteristics, these methods are inefficient: they consume substantial memory, computation, and communication overheads for storing, exchanging, and validating certificates.

Table 11. A comparison between DTLS-based schemes. A security property can have three different values: high (★★★), medium (★★), and low (★). A value is assigned to a security property based on a DTLS scheme's performance to support that property.

Scheme	Interoperability	Resilience	Scalability	Communication	Computation	Memory
Certificate based [128]	★★★	★★★	★★★	★	★	★
DTLS-PSK [122]	★★	★	★	★★★	★★★	★★
Modified DTLS [130]	★	★★★	★★★	★★★	★★	★★
Delegation based [131,132]	★★★	★	★	★★★	★★★	★★

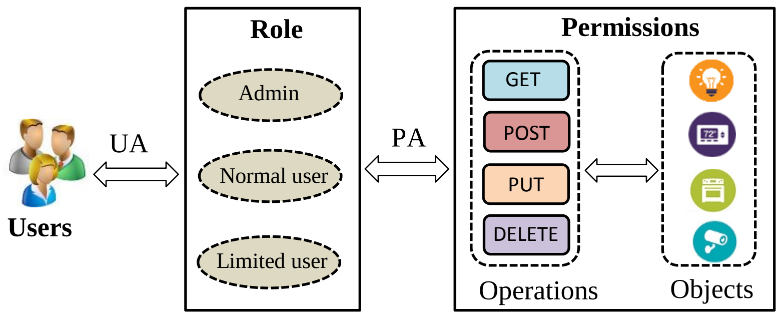
6.3. Service Security

IoT systems are designed based on the producer–consumer model. In an IoT network, smart devices operate as service providers, and end users, such as the smartphone of a device owner, are considered service consumers. For instance, in a smart healthcare service, medical sensors provide services that allow physicians to monitor patients' health conditions remotely using smartphones or through a web portal. Access to these services should be protected to ensure privacy and confidentiality of the sensitive information.

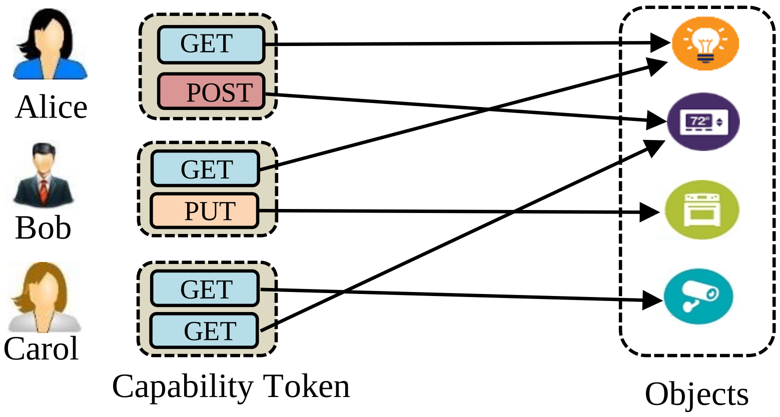
6.3.1. Access-Control Models

Access-control mechanisms aim to achieve two goals: *(i)* authorized users are given access to sensitive information stored on IoT devices; *(ii)* authorized users are allowed to perform only authorized operations, such as issuing commands to smart devices. Access-control models are designed based on three attributes: subjects (users), objects (resources and services), and operations (actions). The subjects are the users or consumers of an IoT system. Resources and services provided by an IoT device are considered as objects. An object's actions toward other objects are defined as operations. An access-control method defines policies that determine an object's rights (or permissions) to perform operations toward objects.

The existing access-control methods can be classified into two major categories: Role-based Access Control (RBAC) [162] and Capability-based Access Control (CapBAC) [163]. In RBAC, every entity of an IoT system, such as a user or device, is assigned a role. Next, permissions to perform certain operations on IoT services and resources are mapped to the role. Figure 19a presents an overview of the RBAC model. Let us consider that two types of roles are defined for a smart home: administrative role and guest role. The administrative role is privileged to perform all sorts of operations, such as read and write, on the smart home appliances. In contrast, the guest role is allowed to perform a particular operation, such as a read request, on certain appliances. Alice, the owner of a smart home, is assigned an administrative role; therefore, she can control all the smart devices of the home. On the other hand, Bob, Alice's partner, may not have access to the home security system as he is assigned with the guest role.



(**a**) RBAC model. UA = User Assignment. PA = Permission Assignment.



(**b**) CapBAC model.

Figure 19. RBAC vs. CapBAC model.

In the CapBAC, every entity of an IoT system is given a capability token that cannot be forged and specifies access rights of the entity. An entity's rights to perform operations on IoT services are defined in its capability token, as shown in Figure 19b. An entity sends a request to an IoT device and attaches its capability token with the request. An object (an IoT device) first verifies the authenticity of the token. Next, it evaluates the entity's capabilities to perform the requested operation. Finally, the object accepts or denies the request according to the verification result.

6.3.2. Access-Control Architectures

Two types of architecture are used to implement RBAC and CapBAC authorization models: centralized architecture and distributed architecture. In the centralized approach, a central entity, such as a cloud server or service, implements access-control logic. The central entity filters out access requests based on authorization policies. The central entity provides various authorization services, such as token verification services, which are instantiated by the Gateway of an IoT network. As shown in Figure 20, a Gateway contacts the central entity to determine whether a user has the rights to perform an action.

In the distributed approach, IoT devices implement the authorization logic. An IoT device evaluates access-control policies before it serves a request. The device accepts or rejects a request based on the authorization policies defined for the requester. Figure 20 presents the policy evaluation process. In the following sections, we survey existing authorization methods designed for IoT systems.

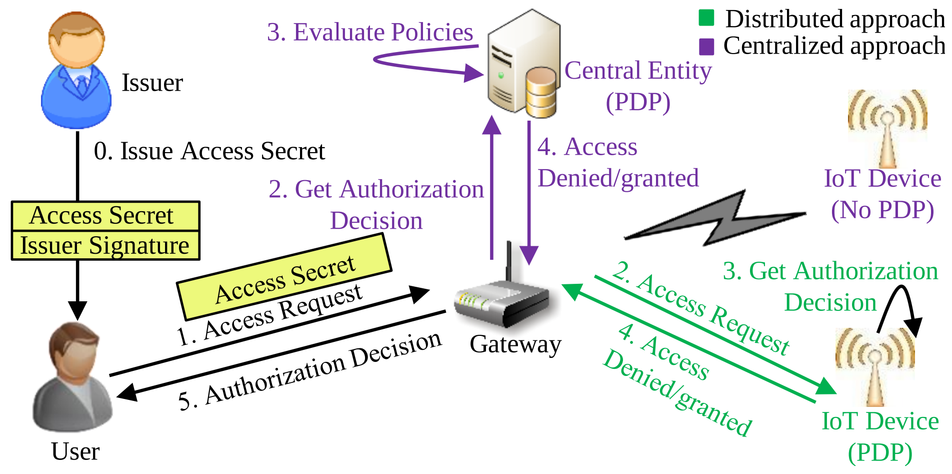


Figure 20. Access-control approach. Policy Decision Point (PDP) is a service that makes authorization decisions. PDP is either offered by an external entity or embedded into an IoT device.

6.3.3. Role-based Access Control

Centralized RBAC: Jing et al. presented an RBAC model to define access-control policies for IoT devices in [134]. The research work also provides methods to manage access and validate authorization policies based on OpenID [164] technology. The proposed model uses trusted central entities for authorization purposes. The applicability of the proposed model cannot be ensured because the authors did not provide any proof of concept implementation of the proposed system.

Context-aware RBAC: An RBAC model with context information is presented in [135]. The proposed model grants permission to a user based on the access polices defined for a requester. During policy evaluation, the proposed system also considers a set of contextual information, such as device status, device or user locations, and time of access, collected from the environment of the service device and requester. The authorization model assigns appropriate permissions to a role according to the characteristics and contextual information of the smart devices, as well as the functions of the role.

6.3.4. Capability-Based Access Control

Centralized CapBAC: Sergio et al. [136] proposed a CapBAC framework for IoT systems. The proposed framework introduces a cloud entity named Policy Decision Point (PDP) that manages the assignment of capability tokens to the service consumers, such as users and client IoT devices. The PDP also validates a capability token to determine whether a consumer is privileged to access certain IoT services. In the proposed system, an IoT device acts as a Policy Enforcement Point. A consumer makes a service request to a service provider and attaches its capability token with the request. The service provider (an IoT device) forwards the capability token to the PDP. The PDP evaluates the capability token against the access policies defined for the consumes and decides whether the consumer will be given access to the requested service.

An assertion-based authorization framework was proposed in [137]. The proposed system utilizes XACML [165] and SAML [165] for authorization process and assertions, respectively. The SAML providers a large number of authorization features. The adoption of all the features can introduce communication overheads for delivering the assertions. To simplify the process for IoT devices, the framework defines a subset of the SAML standards and proposes message formats to encode assertions in JSON [166]. The evaluation of the XACML policies can be a resource-intensive task for IoT devices. Therefore, the proposed system outsources most of the tasks involved in making authorization decisions to a backend authorization servers and have end devices perform only the authorization enforcement.

Pablo Punal et al. [138] proposed an authorization framework based on Kerberos [167] and RADIUS [168]. In the proposed framework, a cloud service provides authentication, authorization, and account management for users and devices of an IoT system. However, the applicability of the proposed framework to IoT environments cannot be confirmed because the authors do not provide an evaluation of the framework in terms of resource efficiency.

Distributed CapBAC: Ramos et al. [139] proposed a CapBAC framework based on the distributed PDP principle. The proposed framework considers a distributed approach without intervention of external entities, such as central authorization servers and services. The distributed architecture enables smart devices to be embedded with all the access-control logics. The authorization process is performed by an IoT device that evaluates the access-control logics defined for a requester and accepts or rejects a request based on the evaluation result. The proposed framework also provides support for authorization delegation and traceability of the access.

The authorization scheme [140] is also based on the distributed CapBAC model. A user or client device requests a service and attaches an access ticket with the request. The user's ability to access the requested service is included in the access ticket. The ticket also includes a cryptographic message digest that is used determine the authenticity of the capabilities. In the proposed model, capabilities are stored in a smart device and the device performs the authorization process, such as policy evaluation and capability verification.

To restrict communications to authorized users, a delegation-based authorization framework based on the CapBAC model is proposed in [141]. In the proposed framework, a Delegation Server, preferably an IoT Gateway, performs the authorization process for an IoT device. The Delegation Server detects a connection request from a remote user to an IoT device and authenticates the user. The access control policies for the user are defined in a database. After authentication, the Delegation Server checks the database and ensures that the user is authorized to make a request to the device. The Delegation Server rejects a connection request from an authenticated but unauthorized user. The Delegation Server does not even notify the IoT device about the unauthorized request. Thus, the proposed framework eliminates request processing and resource allocation overheads from an IoT device. However, the proposed framework only deals with the connection setup authorization and does not address the application layer authorization issues, such as a user's right to access to certain services.

Context-Aware CapBAC: A context-aware CapBAC for a federated IoT network is presented in [142]. The authorization framework provides a unified authorization framework for the IoT system, which forms a network federation by integrating multiple IoT domains, such as a smart home, car, and medical service. In the proposed system, a Federation Manager (IoT-FM) enables authorization delegation in a federated IoT network as well as ensures authentication and authorization for each domain included in the federation. The IoT-FM authorizes a delegation request from a delegator, such as a service consumer in domain A, and grants it to a delegatee, such as a consumer in domain B, who acts on behalf of the consumer in A domain. Although the proposed model can be useful for federated IoT systems, its usability cannot be validated as the details on the implementation and evaluation of the authorization model are not provided.

OAuth Compliant CapBAC: Cirani et al. [143] utilized the OAuth technology [169] to propose an authorization service (IoT-OAS) for the IoT. In IoT-OAS, IoT devices do not implement authorization logics; instead, the authorization functionalities are delegated to an OAuth-based cloud service. Service devices receive an access request from the users or client devices and invoke the authorization service to make decision on the request.

Gerdes et al. [144] proposed a delegation-based authorization framework (DCAF) for the IoT. The architecture and operation model of the DCAF is very similar to the IoT-OAS. The DCAF architecture introduces external authorization servers to reduce the overhead for storing authorization logics and credentials on the device storage. Smart devices do not have to store a large amount of information because information is stored in the servers. The DCAF also reduces the overhead for making authorization decisions by delegating the authorization tasks to an external entity. The DCAF is lightweight compared to the IoT-OAS as it adopts a delegation-based architecture to make authorization decisions.

6.3.5. Analysis and Comparison

With the yardsticks provided in Table 7, we compare the performances of authorization schemes and present the results in Table 12. We noticed that centralized authorization approaches experience reduced computation overhead, offer better interoperability, and simplify the associated access-control managements. These approaches reduce computation overhead by empowering resource-constrained devices to offload costly operations including policy assessment and verification of token status (e.g., verification of signature and ticket validity) to outside entities or proxies. Furthermore, as the contained devices are prevented from storing access policies, issuing secret credentials (e.g., keys or certificates), and even access-control lists, centralized schemes are efficient in terms of memory consumption. However, centralized approaches come with more communication overhead resulting from the information exchanges with an external entity of interest; an IoT device forwards the access token to the assigned external entity and gets an authorization decision in return. Moreover, such communications are also responsible for further delaying the overall response time associated with the respective request, which is not a desirable feature in time-sensitive IoT applications. For example, in an IoT-based healthcare system, a physician sends a request to a wearable medical device to access the latest physiological data of patients, and the request must be served immediately.

The distributed approaches can be suitable for real-time IoT applications because the devices themselves make the authorization decision. While distributed approaches offer a benefit of desirable scalability, it becomes complicated to manage policies in the distributed devices. The interoperability between devices can be affected if the policies cannot be updated on time. The distributed approaches are also labeled underperformers from the perspective of memory efficiency, since the devices themselves need to store the associated policies, secret context, and decision rules. It is important to highlight that integrating CapBAC with a straightforward attribute-based access control approach proves to be effective in Universal Plug-and-Play (UPnP)-enabled IoT systems. This combination facilitates a more nuanced and decentralized solution, enhancing the overall system's efficiency and security [170].

Our analysis finds that very few of the authorization schemes consider the context information for decision making. However, context information can make the decision process easier and faster, because the decision algorithm can jump to a quick resolution by only looking at the context information, such as time of access, device status, user or device location, and so forth.

Table 12. A comparison between authorization methods. Notations used for the security properties are: I = Interoperability, S = Scalability, R = Resiliency, CO = Communication Overhead, MR = Memory Requirement, and CC = Computation Complexity. A security property can have three difference values: high (★★★), medium (★★), and low (★). A value is assigned to a security property based on an authorization scheme's performance to support that property.

Model	Scheme	Approach	I	R	S	CO	CC	MR
RBAC	OpenID based [134]	Centralized	★★★	★★	★★	★★	★★★	★★★
	Context aware [135]	Centralized	★★★	★★	★★	★★	★★★	★★★
CapBAC	Cloud PDP [136]	Centralized	★★	★★	★★	★★	★★	★★
	Embedded PDP [139,140]	Distributed	★★	★★★	★★★	★★★	★★	★
	XACML, SAML based [137]	Centralized	★★★	★★	★★	★★	★★★	★★
	Kerberos, RADIUS based [138]	Centralized	★★★	★★	★★	★★★	★★	★★
	Proxy Assisted [141]	Distributed	★★	★	★	★★★	★★★	★★★
	Context-aware [142]	Centralized	★★	★★	★★	★★	★★★	★★★
	OAuth based [143,144]	Centralized	★★★	★★	★★	★	★★★	★★★

6.4. Summary and Insight

In this section, we have discussed various solutions that can protect IoT devices from physical attacks, ensure the authenticity and confidentiality of communications, and protect security-critical services from unauthorized access. We have found that the TPM technologies, secure boot routines, one-time programmable memories, and secure debugging interfaces can be integrated with IoT hardware and software to prevent host-based attacks. We have also found that the lightweight HIP-based schemes, such as HIP-PSK and Slimfit, can be suitable for managing the identities of billions of IoT devices, including mobile and stationary nodes, and preventing identity impersonation-type attacks. Moreover, we have identified that the delegation-based DTLS schemes can be adopted by IoT systems to ensure confidentiality, authenticity, and integrity of the sensor information as they unburden IoT nodes from computation- and communication-intensive cryptographic operations. Finally, have found from our analysis that the distributed CapBAC models can be utilized, ensuring protected access to security-critical services and enabling IoT devices to provide real-time information at the same time.

Although most of the research we have reviewed modifies the conventional security schemes to make them applicable to IoT systems, only a few of them are tested on the real IoT devices and considered the real-world smart application scenarios. As such, further research is required to validate and confirm the usability of HIP, DTLS, and CapBAC by IoT systems consisting of a large number of smart nodes.

7. Research Directions

The existing security solutions for IoT systems primarily address the access-level security (see Section 5.1) and information security requirements (see Section 5.2). Little attention has been paid to the resource efficiency aspect and functional robustness feature (see Section 5.3) of the security schemes, which are very fundamental requirements for IoT applications. These schemes also hardly consider privacy issues encountered in an IoT-based system. Apart from that, there are many other security aspects that are either not properly addressed or remain unnoticed. This is because many recent and proven technologies, including M2M [46], RFID [171], and Ubicomp [172], are yet to be in complete convergence with the IoT paradigm. In this section, we identify those security problems and try provide some guidelines to design the appropriate solutions for them. Table 13 summarizes the research problems along with the associated research directions. The details of these open issues and future directions are successively presented below.

Table 13. Research problems and guidelines.

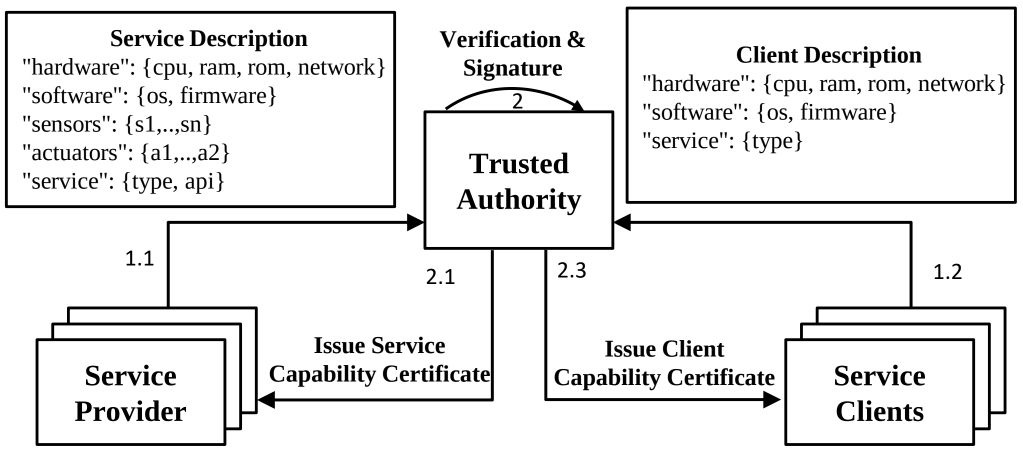
Research Domain	Research Questions	Directions
Service Discovery	How can service impersonation be identified?	Capability-based service advertisement and discovery (Section 7.1)
	How can malicious service discovery requests be identified?	
Identity Privacy	How can movement profiling for mobile IoT devices be prevented?	Use of unique device identifiers from locations-to-locations and sessions-to-sessions (Section 7.2)
	How can communication relations be prevented? How can users control access to their sensor data used by third-party services?	
Data Privacy	How can privacy be preserved for sensor data shared with third-party services for providing personalized services?	A transparency layer can be implemented that will allow users to know who has access to their data and will enable users to determine who can have access to their data (Section 7.3)
	How long will the data remain shared?	
Application Data Security	How can security at the application layer be applied?	Application messages can be encrypted to ensure the confidentiality of exchange messages, while the headers of the application layer protocol should be left unencrypted for protocol translation (Section 7.4)
	How can the confidentiality of application data be ensured during protocol translation?	
	Can existing application layer security schemes be modified for resource-limited IoT devices?	
Software Update	How can software updates be applied to multiple devices simultaneously that are located in the edge or distributed networks?	Resource-efficient software update method that has low computation overheads on IoT device for verifying software authenticity as well as fewer communication overheads for delivering patches over lossy networks (Section 7.5)
	How can smart devices be enabled to verify integrity and authenticity of software updates?	
	How can IoT operating systems be enabled to receive new software updates?	
Credential Security	How can credentials stored on device memory be protected from memory probing?	Security schemes can be designed based on Physically Uncloneable Functions to ensure the security of the credentials embedded with smart devices (Section 7.6)
	How can confidentiality of credentials stored on device memory be ensured?	
Network Security	How can anomalies on IoT networks be detected?	Machine-learning-based models to identify new threats and attacks (Section 7.8)
	How can the knowledge base remain updated for identifying new or unseen network attacks?	
	How can the optimum level of control and monitoring on packets be achieved?	
Digital Forensics	How can evidence collection be enabled in the IoT environment?	A Blockchain-based distributed and decentralized network to maintain a chain of custody of the evidence and avoid single points of failures on storage media (Section 7.9)
	How can secure provenance of the evidence be maintained?	
	How can verification of authenticity and integrity of evidence during an investigation be enabled?	

7.1. Secure Service Discovery

Smart devices can be used to perform service impersonation attacks. A malicious device can provide certain services although it does not have the required software or hardware. An adversary can configure a smart refrigerator to present itself as a medical sensor or thermostat. The goal of the adversary is to provide fake information to client devices. Similarly, in the client impersonation attack, a client device can search for particular services, although it does not have capabilities to interact with the services, and understand information returned by a service device. One of the goals of an attacker is to force target devices to process discovery messages. A mobile application that is designed to interact with smart home appliances can be malicious to discover and communicate with medical sensors located in the home network. The mobile application can send malformed requests to the medical sensors. The processing of the malicious discovery messages and malformed requests can result in resource exhaustion, such as shorter battery life, and service unavailability.

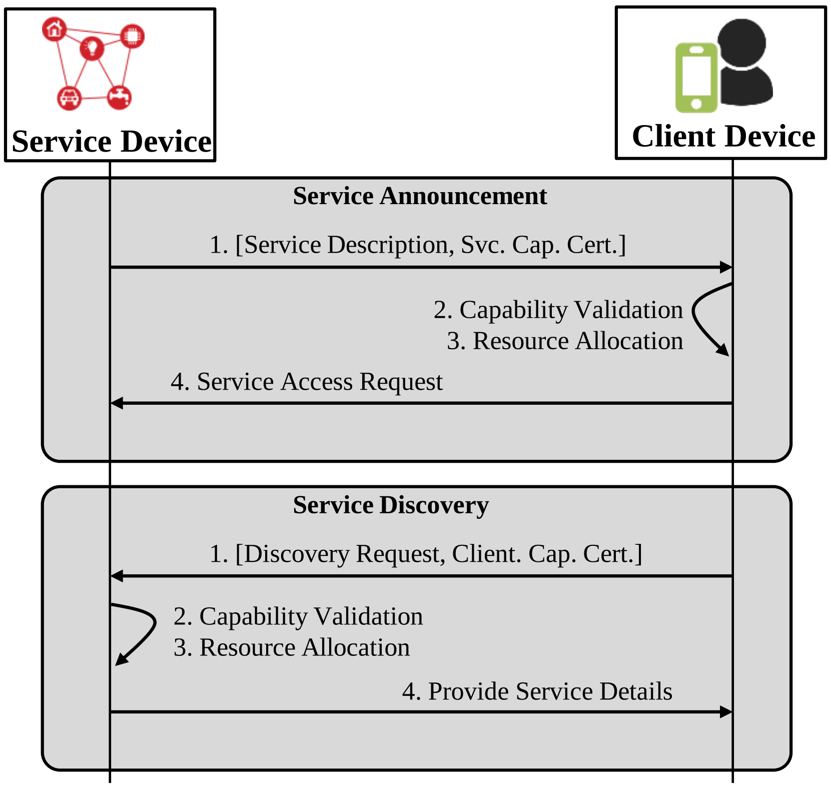
A service authentication scheme can be designed to prevent service devices from offering fabricated services and client devices from discovering unauthorized services. Figure 21 shows an overview of such a secure service discovery model. A trusted entity, such as a certificate authority, issues service capability certificates (SCC) and client capability certificates (CCC) to service and client devices, respectively. The SCC and CCC contain the signature of the trusted party; therefore, they cannot be forged. The SCC contains device and service specifications of a service device. The device specification provides information on sensors and actuators embedded with the service device. The service specification includes a list of services the device is authorized to provide as well as a list of web interfaces, such as APIs, the device is authorized to expose for exchanging information. Similarly, a CCC issued to a client device contains information on the types of services the client is allowed to interact with.

A service device must attach its SCC to its service announcement message. A client device does not respond to a service announcement message that contains service information not included in the service device's SCC. Thus, the client device avoids allocating resources, such as memory, storage, and CPU cycles, for storing and processing service information. Likewise, a client device has to attach its CCC with a service discovery message. A service device does not process a discovery request made by a client device if it finds that the client device is not capable of consuming the requested service as mentioned in its CCC. Hence, the verification of service and client devices' capabilities can enable IoT devices to prevent service and client impersonation attacks.



(**a**) Device enrollment.

Figure 21. *Cont.*



(**b**) Discovery and announcement.

Figure 21. A model for secure service discovery.

7.2. Identity Privacy

IoT networks are dynamic due to the mobility of some smart devices, such as wearable devices and smart vehicles. Every device of an IoT system must be identified uniquely. The identifiers of these devices should be cryptographically proctored to defend against impersonation-type attacks. Moreover, the identifiers, such as host identities [117] and certificates [128], of a device used for authentication should be unique from locations to locations, sessions to sessions, and networks to networks. The use of static identifiers makes IoT devices and device owners vulnerable to cyber espionage and user-targeted attacks, such as movement profiling (Figure 22) and communication relation (Figure 23).

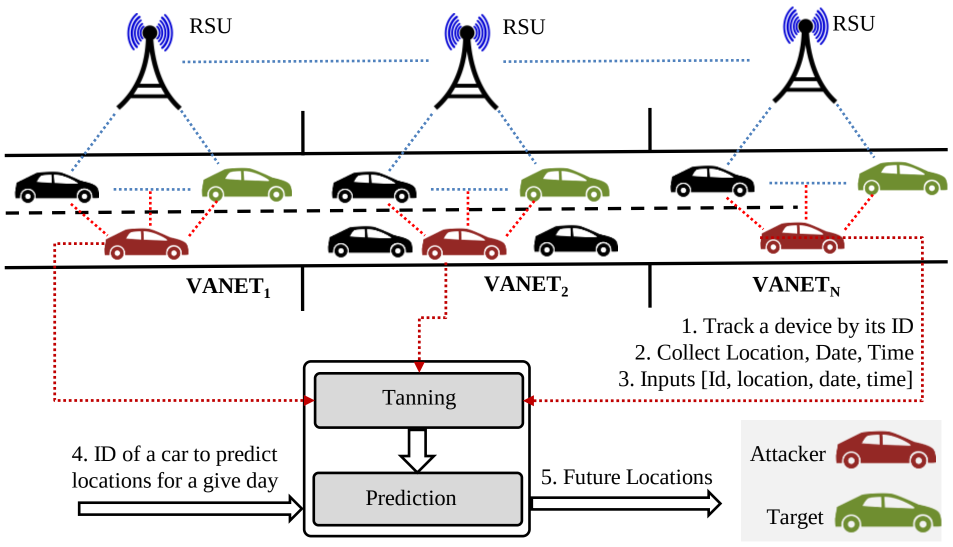

RSU
RSU
RSU
VANET$_1$
VANET$_2$
VANET$_N$
1. Track a device by its ID
2. Collect Location, Date, Time
3. Inputs [Id, location, date, time]
Tanning
Prediction
4. ID of a car to predict locations for a give day
5. Future Locations
Attacker
Target

Figure 22. An adversary profiles the movement of a target by tracking the static identifier of the target. RSU = Roadside Unit. VANET = Vehicular Ad Hoc Network.

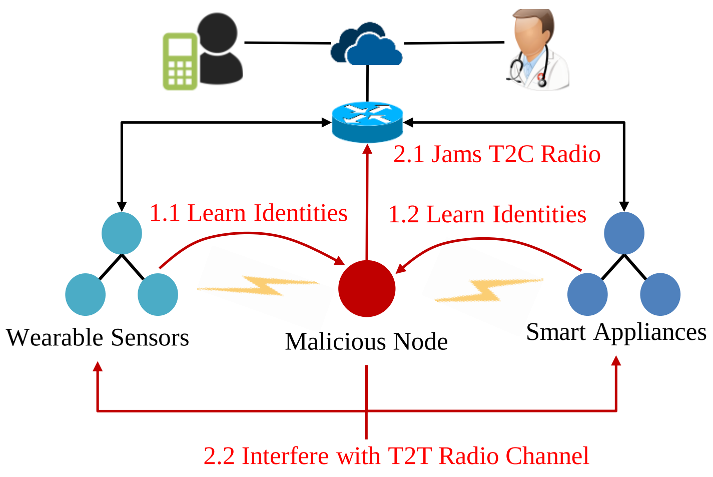


Figure 23. An adversary tracks the identifiers of target devices for disrupting communications. T2T = Things to Things, T2C = Things to Clouds.

As shown in Figure 22, an adversary can learn the locations of a smart car and its owner by tracking the identifier of the car used for network authentication (when the car joins a network) and service authentication (when the car communicates with other cars and roadside services). Similarly, adversaries being co-located with victim devices can record device identifiers during device-to-device communications for device targeted attacks. For instance, an adversary can disrupt communication channels when it identifies that a target device starts communication with other devices and services. Figure 23 shows such an attack.

To protect location and communication privacy, IoT devices should be enabled to compute or use a one-time device identifier, which is similar to the one-time password [173], for network authentication and device-to-device mutual authentication. Authentication schemes based on Zero Knowledge Proof algorithms [174,175] can be adopted to preserve identity privacy.

7.3. Data Privacy

An IoT service provider can utilize third-party services, such as Big Data analysis services, to analyze sensor information. Based on the analysis results, the service provider can provide various analytics to the users. For instance, the manufacturers of wearable medical sensors and fitness trackers can share users' health data with hospitals and pharmaceutical companies to learn users' daily activities and suggest prescriptions to the users accordingly. Similarly, the service provider of a smart home can share data collected from smart appliances, such as thermostats, refrigerators, washing machines, cameras, and so forth, with a third-party service to better understand the context of the smart home. However, sharing information with third-party services can be a threat to the privacy and confidentiality of user and sensor information. Therefore, the owners and users of the smart devices should be allowed to verify that the confidentiality of their data is not compromised during data collection, sharing, and collaboration phases.

A data transparency service (DTS) can be implemented in IoT systems to protect data privacy. As shown in Figure 24, a DTS will allow the owners to know who has access to their data. DTS will also enable the owners to decide the consumers of the data. Moreover, DTS will let owners validate data genuineness, such as the origin of the data and data freshness.

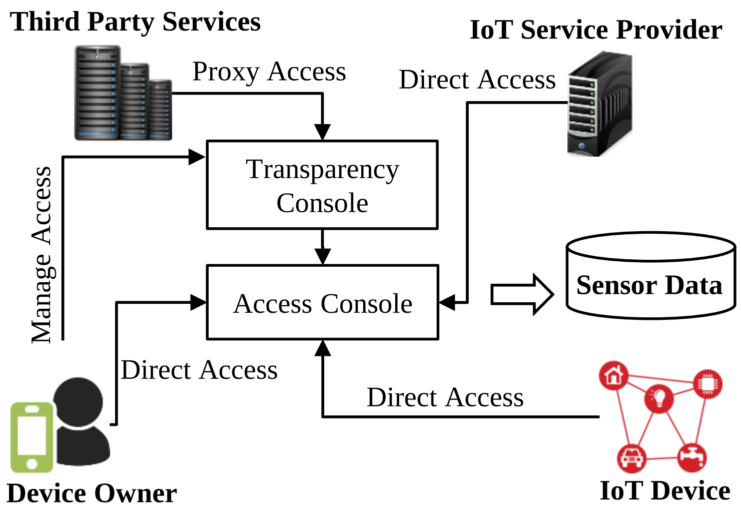


Figure 24. An overview of the Data Transparency Service.

7.4. Application Data Security

IoT devices operate on lossy networks and use IPv6 addresses and CoAP to communicate among themselves. However, these devices communicate to IoT service providers located in the Cloud using IPv4 addresses and HTTP. As shown in Figure 25, a Gateway device located between IoT devices and IoT Cloud services enables communications between IoT devices and the Cloud service by translating one protocol to another, such as IPv4 to IPv6, or HTTP to CoAP, and vice versa. However, the Gateway device can learn the contents of application payloads during the protocol translation.

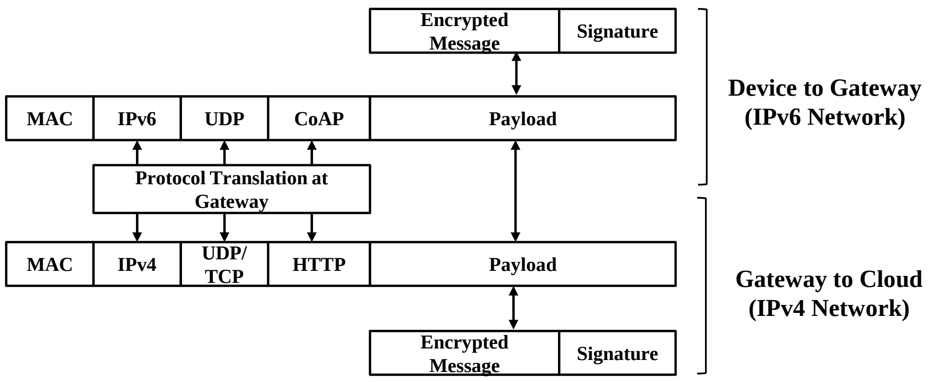


Figure 25. Application data security during protocol translation.

Security at the application layer can prevent the protocol translators, such as Gateways, from learning application payloads. The application data security also simplifies the security requirements for the underlying layers. It eliminates the overheads for providing security for every packet at the underlying layers, such as Transport and Network layers, because only applications data needs to be secured. Figure 25 presents a process to ensure the security and privacy of application data. Unlike the Transport Layer Security that encrypts both headers and payloads at the session presentation layer, only payloads are encrypted at the application layer. Therefore, application data are not exposed to the Gateway when it handles and processes messages exchanged between IoT devices and clients.

S/MIME [176] and SRTP [177] standards can be utilized for application data security. However, further research is required to apply S/MIME and SRTP to the CoAP-enabled IoT devices and networks because these security standards are not primarily designed for resource-limited environments. The standards may need to be modified to make them suitable for IoT systems.

7.5. Software Update

An IoT network includes heterogeneous smart devices from various manufacturers. Devices are installed with software with various security configurations. Installed software must be updated periodically with new security patches; however, it may not be possible to apply software updates to IoT devices because of the following properties of the IoT components:

1. Smart devices can be configured such that they are only accessible from the local network. A user has to be co-located with a device in the same network to get access to it through the Gateway. In such scenarios, security updates cannot be applied directly as the devices are not connected to the Internet.
2. IoT operating systems may not have security modules to receive and integrate software updates and security patches. For instance, security updates cannot be applied to Contiki [32], RIoT [33], and TinyOS [178] operating systems, as they do not have software update methods.
3. There may have a significant communication overhead for delivering software updates over the lossy and limited-bandwidth networks, such as 6LoWPAN [68] and Zigbee [69]. Software updates need to be sent in multiple fragments. Some of the fragments may need to be retransmitted because of the lossy networks. However, the networking protocols 6LoWPAN and Zigbee do not provide a mechanism for retransmitting the missing fragments. An entire message has to be retransmitted when one or more fragments are lost.
4. Devices may not have enough memory to store software updates for verifying the authenticity and integrity of the updates.
5. Simultaneous software updates may need to be applied to the devices of an IoT system to maintain interoperability between these devices.

A resource-efficient Software-Update-as-a-Service (SUaaS) can be designed to apply security updates to IoT devices that are located in the lossy networks and may not be connected to the Internet. We present an overview of the SUaaS in Figure 26. The SUaaS has two components: cloud and edge. The cloud component comprises three modules: artifactory, security, and distribution. The artifcatory module stores the new version of the software or security patches. Device manufacturers and service providers publish software updates on the artifactory. The security module verifies the authenticity and integrity of the software to ensure that trusted publishers published the updates. The security module also performs static and dynamic code analysis to ensure that the published software does not contain signatures of the malicious codes, such as malware. The distribution module sends the software updates to the edge component.

An Edge router located in the edge network acts as an edge component. The Edge router fragments software into multiple fragments that can fit in the MTU of the communication link. Next, the Edge router computes an aggregated signature for the fragments, avoiding appending a signature to every fragment and reducing the total number of bytes required to deliver over the lossy network. To this end, the Edge router multi-casts the fragments to the lossy network for simultaneous software update.

A lightweight aggregated-signature verification scheme based on a Bloom filter [179] can be used such that resource-constrained devices can verify the authenticity and integrity of the individual piece of the software with minimal computation costs. In the conventional security method, the Edge router appends signatures with the fragments ($F_i||Sig_i$) and devices verify the signatures Sig_i after fragments. In the Bloom filter-based aggregated-signature, the Edge router computes hashes of the fragments ($HASH(F_i)$) and sets the corresponding bits in the Bloom filter (BF_x), which is an array of X bits, according to the hash results. The Edge router only signs the BF_x as $BF_x||Sig_{BF}$ and sends it to the devices. The devices only validate the Sig_{BF} and ensures that the corresponding bits of a hash of a fragment $HASH(F_i)$ is set in the BF_x. Hence, a device can avoid computation cost for signature validation for each software fragment.

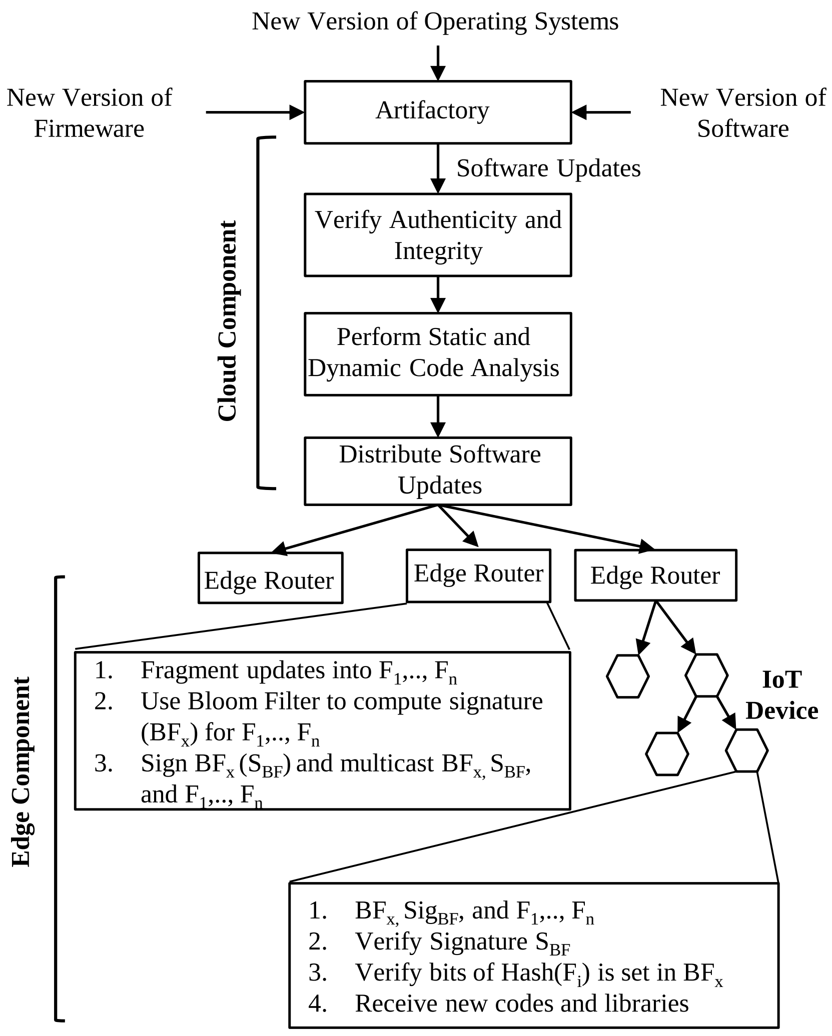

New Version of Operating Systems
New Version of Firmware
Artifactory
New Version of Software
Software Updates
Cloud Component
Verify Authenticity and Integrity
Perform Static and Dynamic Code Analysis
Distribute Software Updates
Edge Router
Edge Router
Edge Router
Edge Component
1. Fragment updates into F_1,.., F_n
2. Use Bloom Filter to compute signature (BF_x) for F_1,.., F_n
3. Sign BF_x (S_{BF}) and multicast $BF_{x,}$ S_{BF}, and F_1,.., F_n
IoT Device
1. $BF_{x,}$ Sig_{BF}, and F_1,.., F_n
2. Verify Signature S_{BF}
3. Verify bits of Hash(F_i) is set in BF_x
4. Receive new codes and libraries

Figure 26. Secure software update scheme.

7.6. On-Device Credential Security

Smart devices store certificates and public, private, and shared keys on their memories. Devices can be a subject to node-probing attacks if they are deployed in remote locations and left unattended. Adversaries can tamper with device memories to extract credentials. Later, these credentials can be used to impersonate real devices.

A security scheme can be designed based on the Physically Uncloneable Function (PUF) [180–183] to store keying materials securely. A PUF is a circuitry that can be embedded with IoT devices. The PUF is considered to be the digital fingerprint for a device, is unique for a device, and cannot be cloned. A PUF circuit takes challenge bits (e.g., a bit stream of 128 bytes) as inputs and computes response bits for the challenge. Multiple devices embedded with a same PUF compute unique responses for the same challenge. Therefore, a device can encrypt its cryptographic materials using the PUF response to a challenge and then store the encrypted credential in its memory.

Figure 27 presents an overview of the scheme that stores credentials securely on the device's storage. A challenge is stored in the device memory in plain text. This challenge is given as the input to a PUF circuit. The PUF computes a response for the challenge, and the response is used as the key of a symmetric-key encryption algorithm to encrypt credentials. Similarly, the response is used to decrypt the encrypted credentials.

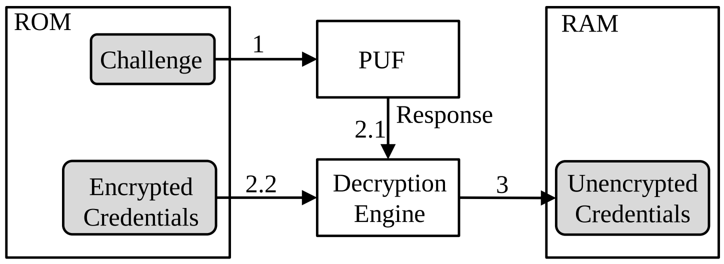


Figure 27. A PUF-based scheme for storing credentials securely.

7.7. Memory-Aware Security Solutions

As mentioned in the early part of this paper, memory efficiency is crucial in resource-limited IoT devices because these devices often have constraints in terms of processing power, storage, and energy. Optimizing security schemes to reduce memory consumption is essential to ensure that security measures do not compromise the functionality and performance of the IoT devices. Here are some controls and strategies to address memory efficiency in resource-limited IoT devices:

Cryptographic Algorithm Selection: We can choose lightweight cryptographic algorithms that require less memory and processing power. For example, consider using elliptic curve cryptography (ECC) instead of RSA for public key cryptography, as ECC generally requires smaller key sizes and less computational resources [184].

Optimized Libraries and Code Size: It would be advantageous to utilize memory optimization techniques such as code and data compression to reduce the overall memory footprint [185,186]. This may involve using tools and compilers that can optimize code for size. We can use lightweight cryptographic libraries that are specifically designed for resource-constrained environments [127]. These libraries are often tailored to minimize memory usage while still providing essential security functions. It would be useful to employ code size optimization techniques during the development and compilation process. This includes removing unnecessary code, using compiler optimizations, and employing techniques such as function inlining.

Dynamic Memory Allocation: We need to minimize or avoid dynamic memory allocation (e.g., using malloc in C) [187], as it can lead to fragmentation and increased memory overhead. Instead, prefer static memory allocation when feasible.

Secure Key Storage: It is important to implement secure key storage mechanisms that use minimal space while ensuring the confidentiality and integrity of cryptographic keys. We also need to consider hardware-based security solutions or trusted execution environments when available. It is recommended to optimize the storage of cryptographic materials and program codes in flash memory. This can involve efficient data structures and storage formats to minimize the space required for storing keys, certificates, and other cryptographic parameters [188].

Hardware Acceleration: We can exploit hardware-based cryptographic accelerators if available on the IoT device. Offloading cryptographic operations to dedicated hardware can significantly reduce the burden on the main processor and conserve memory [189,190].

Firmware Over-the-Air (OTA) Updates: We can also suggest implementing efficient firmware update mechanisms [191,192] that only transmit and store the necessary changes, which reduces the overall data transfer and storage requirements.

7.8. Network Anomaly Detection

Authentication and encryption techniques protect end-to-end communications. However, adversaries can perform various types of networks attacks, such as SQL injection, Cross-Site Scripting, Malware injection in payloads, Denial of Service, and Distributed Denial of Service attacks, from inside the network and the Internet. These types of wireless attacks can be prevented by deploying Intrusion Prevention Systems (IPS) [193,194] to the IoT systems. It is also possible to detect the wireless attacks by using Intrusion Detection Systems (IDS) [193,194]. The IPS and IDS capture network traffic and perform deep packet analysis to identify network anomalies. An IoT network can achieve a higher degree of security against network attacks by increasing control and monitoring on the network packets. However, the increase in the inspection of network packets can be a threat to the privacy of the user and sensor information. As a result, there are opportunities to research anomaly detection systems that apply an optimal level of security control on the network traffic and can identify network intrusions effectively without compromising information privacy.

Moreover, the IoT application domains are still emerging. New smart applications and services can introduce unseen and unpredictable vulnerabilities, threats, and attacks in IoT systems. Therefore, research can be done to enable anomaly-detection services to identify and mitigate emerging network attacks based on the knowledge base of existing threats.

Machine learning techniques can be adopted to identify new and emerging network anomalies. Figure 28 shows an overview of an adaptive security model to detect current and emerging network attacks. The IPS and IDS receive incoming and outgoing packets and inspect the payloads to find a match with anomaly signatures or rules. The IPS and IDS generate alter when they find matches. However, the database for signatures and rules needs to be updated periodically to detect most recent attacks. An attack has to be seen first to identify its signatures. Machine learning models can be developed that inspect network packets, predict new attacks, and create signatures for alerts. As shown in Figure 28, a Network Monitoring System (NMS) [195] captures all the network traffic, parses it to generate session information (such as protocols, headers, and payloads), and stores the session information in a database. A machine learning model uses the session information or metadata to identify unusual or suspicious communications and messages and then creates signatures using the metadata. Hence, new rules can be added to the signature database.

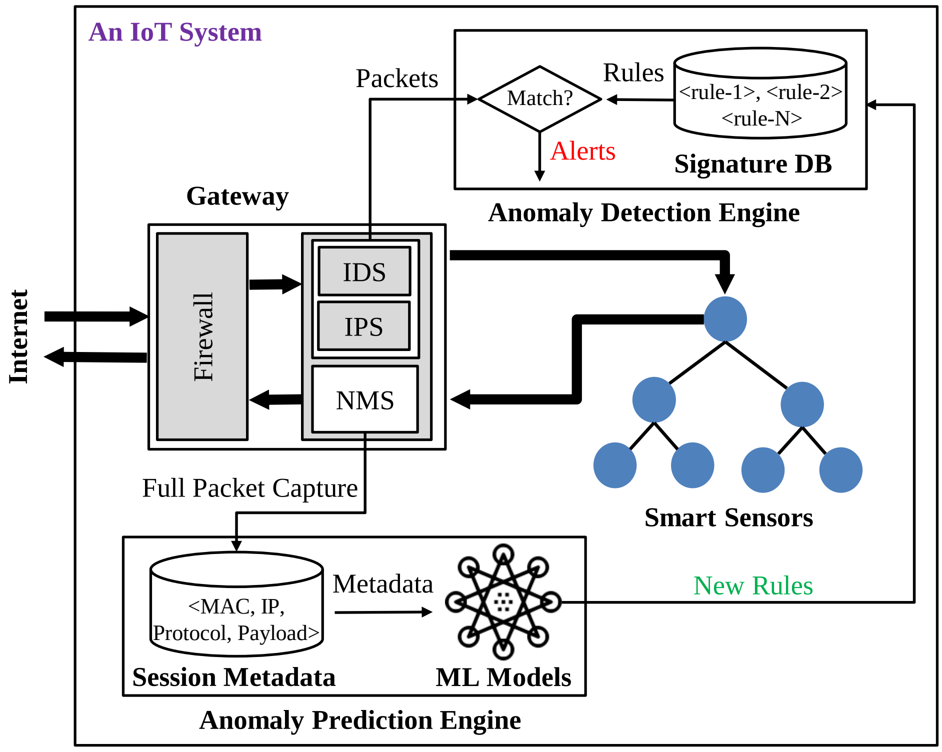


Figure 28. An adaptive anomaly-detection system. NMS = Network Monitoring System. ML = Machine Learning.

7.9. IoT Forensics

In the IoT environment, smart devices, applications, and communications can be used as tools for committing cyber-crimes. For example, it can be life threatening if an adversary compromises an IoT-enabled pacemaker. To investigate such criminal cases, investigators have to execute a cyber forensic process and determine the facts about an incident [196]. The forensic process, at first, identifies incidents and evidence in an IoT-based system and then stores the evidence securely. An investigator collects the evidence and generates proofs. However, traditional forensic tools and techniques, such as media [197], cloud [198], and network forensics [199], cannot be utilized to investigate IoT-based cyber-crimes because they are not designed for digital systems that are mobile and consist of a large number of devices. Some of the limitations of the existing forensic tools are as follows.

1. Media forensics must have physical access to the storage of a digital device. It may not be possible to retrieve logs stored in the memory of a medical sensor, which is required to remain online and implanted in a patient's body.
2. Cloud forensics analyze logs of the cloud services that run on the cloud servers. The cloud logs may not be used as evidence for investigating incidents that occur in the edge networks where IoT services run on smart devices and are accessed locally.
3. Network forensics may not be suitable for analyzing incidents in the smart systems where devices are mobile and network topologies change over the time, such as ad hoc networks.

Although the current research [200,201] on IoT forensics highlight the sources of evidence in IoT and provide directions on incident examination procedures, there are many aspects of IoT forensics to research further. Researchers can take opportunities to define evidence in the IoT-based systems, develop scalable storage mechanisms to log large amounts of evidence, provide methods to generate secure provenance of the evidence, and propose techniques to analyze evidence.

As shown in Figure 29, a distributed and decentralized Blockchain network, which is similar to the Bitcoin network, can be used to store evidence securely. The interactions between the IoT entities, as shown in Figure 4, can be considered as the evidence. These interactions can be collected and stored in a public digital ledger that is similar to the payment transactions in the Bitcoin network. The interactions can be grouped to form transaction blocks. The parties involved in communication have to sign the transaction so that they cannot deny their participation in an incident related to cyber-crime.

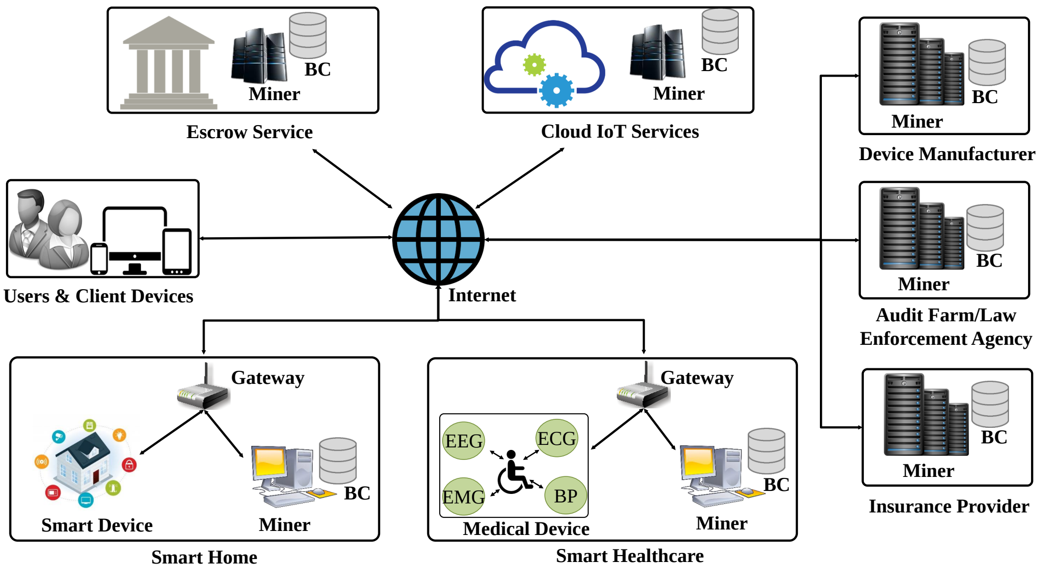


Figure 29. A Blockchain-based forensic framework.

The miners of the blockchain network and copies of the blockchains containing interaction blocks as evidence can be maintained by the stakeholders of an IoT system, such as device manufacturers, service providers, device owners, audit farms, law enforcement agencies, cloud providers, and insurance providers. The evidence ledger will be made publicly available. In the public blockchain, the forensic transactions can be encrypted by the public key of a trusted entity, such as an Escrow service or certificate authority, to ensure the confidentiality of the evidence. During an investigation, an investigator will collect transactions from the public ledger and contact the trusted party to obtain the unencrypted evidence. The investigator will analyze the interactions to establish facts. The transaction of the forensic ledger can be used both for investigation cyber-criminal cases and for resolving disputes in IoT systems, such as violation of service-level agreements by hospitals in smart healthcare systems or traffic laws by smart cars, drivers, and pedestrians in smart transportation systems.

The distributed and decentralized copies of the forensic ledger will help avoid single points of failure on the evidence storage media. Moreover, a single entity cannot tamper with the evidence, such as insertion, modification, and deletion, to change the outcome of an investigation because the evidence ledger is publicly available and distributed among the stakeholders. The distributed and decentralized properties of the forensic ledger will also ensure the high availability of the evidence.

8. Concluding Remarks

In this paper, the recent literature of IoT security has been surveyed and critically analyzed with an emphasis on the following aspects from holistic viewpoint: the basic configurations of IoT networks and components, security vulnerability, attack taxonomy, security requirements, security solutions, and research directions. To obtain some insights into the question of to what extent an attack would allow adversaries to take control of an entire IoT network, we have examined various security vulnerabilities and attack scenarios for IoT systems. For a deeper understanding of the requirements for IoT security schemes, we have categorized the available and possible challenges into three broad groups—access-level security requirements, information security properties, and functional security requirements—and have analyzed each of them. We have provided a comprehensive survey on various security solutions to safeguard IoT devices from physical attacks, ensure the authenticity and confidentiality of communications, and protect security-critical services from unauthorized access. In addition, we have performed an exhaustive analysis on the current solutions and identified issues in them that require further research. Furthermore, we have highlighted numerous emerging perspectives, such as secure service discovery, on-device credential security, network anomaly detection, and IoT forensics, that are poorly addressed or unnoticed and have provided direction for each of them. More application domains can take advantage of the IoT paradigm if these problems are solved. Finally, to overcome the inherent limitations of the conventional forensic approaches such as media, cloud, and network forensics, we have provided guidelines for designing a Blockchain-based investigation forensic framework suitable for a complete IoT infrastructure involving a massive number of devices. In sum, the results of this systematic survey are expected to be useful to researchers working in the area of security and IoT-based systems.

Author Contributions: Conceptualization, M.H. and R.H.; methodology, M.H. and S.M.R.I.; software, G.K. and S.N.; validation, M.H., G.K. and S.M.R.I.; formal analysis, M.H., G.K. and S.M.R.I.; investigation, M.H., G.K., R.H., A.S., S.N. and S.M.R.I.; resources, M.H. and R.H.; writing—original draft preparation, M.H., R.H. and S.M.R.I.; writing—review and editing, G.K., A.S. and S.N.; visualization, M.H., G.K. and S.M.R.I; supervision, R.H. and S.M.R.I.; project administration, G.K., A.S. and S.N.; funding acquisition, R.H. and S.M.R.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are contained in the article itself.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hossain, M.; Islam, S.R.; Ali, F.; Kwak, K.S.; Hasan, R. An Internet of Things-based health prescription assistant and its security system design. *Future Gener. Comput. Syst.* **2018**, *82*, 422–439. [CrossRef]
2. Ali, F.; Islam, S.R.; Kwak, D.; Khan, P.; Ullah, N.; Yoo, S.j.; Kwak, K.S. Type-2 fuzzy ontology-aided recommendation systems for IoT-based healthcare. *Comput. Commun.* **2018**, *119*, 138–155. [CrossRef]
3. Islam, S.R.; Uddin, M.N.; Kwak, K.S. The IoT: Exciting possibilities for bettering lives: Special application scenarios. *IEEE Consum. Electron. Mag.* **2016**, *5*, 49–57. [CrossRef]
4. Islam, S.R.; Kwak, D.; Kabir, M.H.; Hossain, M.; Kwak, K.S. The internet of things for health care: A comprehensive survey. *IEEE Access* **2015**, *3*, 678–708. [CrossRef]
5. O'Neill, M. Insecurity by design: Today's IoT device security problem. *Engineering* **2016**, *2*, 48–49. [CrossRef]
6. Díaz, M.; Martín, C.; Rubio, B. State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. *J. Netw. Comput. Appl.* **2016**, *67*, 99–117. [CrossRef]
7. Zhao, M.; Hu, C.; Song, X.; Zhao, C. Towards dependable and trustworthy outsourced computing: A comprehensive survey and tutorial. *J. Netw. Comput. Appl.* **2019**, *131*, 55–65. [CrossRef]
8. Han, W.; Xue, J.; Wang, Y.; Liu, Z.; Kong, Z. MalInsight: A systematic profiling based malware detection framework. *J. Netw. Comput. Appl.* **2019**, *125*, 236–250. [CrossRef]
9. HP. Internet of Things Research Study. 2014. Available online: https://h41382.www4.hpe.com/gfs-shared/downloads-352.pdf (accessed on 22 January 2024).
10. Kirsten, S. Cross Site Scripting (XSS). Available online: https://owasp.org/www-community/attacks/xss/ (accessed on 22 January 2024).
11. Proofpoint. Proofpoint Uncovers IoT Cyberattack. 2014. Available online: https://www.proofpoint.com/us/proofpoint-uncovers-internet-things-iot-cyberattack (accessed on 22 January 2024).
12. Security, K. Mirai IoT Botnet. 2016. Available online: https://krebsonsecurity.com/2017/12/mirai-iot-botnet-co-authors-plead-guilty/ (accessed on 22 January 2024).
13. Reading, D. Air Force Researchers Plant Rootkit In A PLC. 2014. Available online: http://www.darkreading.com/attacks-breaches/air-force-researchers-plant-rootkit-in-a-plc/d/d-id/1141218? (accessed on 22 January 2024).
14. Times, N. Stuxnet Computer Worm. 2011. Available online: http://www.nytimes.com/2011/01/16/world/middleeast/16stuxnet.html (accessed on 22 January 2024).
15. TechCrunch. BrickerBot: A Vigilante Worm That Destroys Insecure IoT Devices. 2017. Available online: https://techcrunch.com/2017/04/25/brickerbot-is-a-vigilante-worm-that-destroys-insecure-iot-devices/#:~:text=BrickerBot%20finds%20these%20devices%20and,by%20formatting%20the%20internal%20memory (accessed on 22 January 2024).
16. ZDNet. Finns Chilling as DDoS Knocks Out Building Control System. 2016. Available online: https://www.theregister.co.uk/2016/11/09/finns_chilling_as_ddos_knocks_out_building_control_system/ (accessed on 22 January 2024).
17. Labs, I. Car Hacking. 2014. Available online: http://blog.ioactive.com/2014/04/car-hacking-2-content.html (accessed on 8 August 2018).
18. Tech, C. Hacking the Drug Pump. 2015. Available online: http://money.cnn.com/2015/06/10/technology/drug-pump-hack/ (accessed on 22 January 2024).
19. Rapid7. Hacking IoT: A Case Study on Baby Monitor Exposures and Vulnerabilities. 2016. Available online: https://information.rapid7.com/iot-baby-monitor-research.html (accessed on 22 January 2024).
20. Cerrudo, C. Hacking US Traffic Control System. 2014. Available online: https://ioactive.com/hacking-us-and-uk-australia-france-etc/ (accessed on 22 January 2024).
21. Oren, Y.; Keromytis, A.D. From the aether to the ethernet attacking the Internet using broadcast digital Television. In Proceedings of the USENIX Security, San Diego, CA, USA, 20–22 August 2014.
22. Hoque, M.A.; Hossain, M.; Noor, S.; Islam, S.R.; Hasan, R. IoTaaS: Drone-based Internet of Things as a service framework for smart cities. *IEEE Internet Things J.* **2021**, *9*, 12425–12439. [CrossRef]
23. Zolertia. Z1 Mote IoT Device. 2016. Available online: http:///zolertia.sourceforge.net/ (accessed on 22 January 2024).
24. SkyMote. T-Mote Sky Iot Device. 2016. Available online: http://wirelesssensornetworks.weebly.com/1/post/2013/08/tmote-sky.html (accessed on 22 January 2024).
25. Opal. Opal Sensor Node. 2016. Available online: http://www.net.in.tun.de/en/sandbox/wireless-sensor-networks/ (accessed on 12 August 2021).
26. Mote, O. Open Hardware for the Internet of Things. 2016. Available online: http://openmote.com/product/openmote-b-platinum-kit/ (accessed on 12 August 2021).
27. Libelium. Waspmote: The Sensor Device for Internet of Things Developers. 2016. Available online: http://www.libelium.com/products/waspmote/ (accessed on 22 January 2024).
28. Arduino. Arduino Uno: An IoT Development Board. 2017. Available online: https://store.arduino.cc/usa/arduino-uno-rev3 (accessed on 22 January 2024).
29. Arm-Mbed. Mbed: A Development Board for Rapid Prototyping of IoT Applications. 2017. Available online: https://os.mbed.com/platforms/mbed-LPC1768/ (accessed on 22 January 2024).

30. Weptech. A 6LoWPan Border Router. 2017. Available online: https://www.ti.com/document-viewer/lit/html/SSZTBO7 (accessed on 22 January 2024).
31. Weinzierl. KNS Stacks: A Development Board for KNX Applications. 2017. Available online: https://www.weinzierl.de/index.php/en/all-knx/knx-stacks-en/development-hardware-en (accessed on 22 January 2024).
32. Contiki. Contiki OS: An Open Source Operating System for the Internet of Things. 2016. Available online: http://www.contiki-os.org/ (accessed on 22 January 2024).
33. RIoT. RIOT: A Small Operating System for Resouce Constrained Systems. 2017. Available online: https://riot-os.org/ (accessed on 22 January 2024).
34. Kent, S.; Seo, K. Security Architecture for the Internet Protocol. *RFC IETF*. 2005. Available online: https://www.rfc-editor.org/rfc/rfc4301 (accessed on 22 January 2024).
35. Rescorla, E.; Modadugu, N. Datagram Transport Layer Security. *RFC IETF*. 2006. Available online: https://www.rfc-editor.org/rfc/rfc4347.html (accessed on 22 January 2024).
36. Zhou, Z.; Wu, C.; Yang, Z.; Liu, Y. Sensorless sensing with WiFi. *Tsinghua Sci. Technol.* **2015**, *20*, 1–6. [CrossRef]
37. Wang, C.; Jiang, T.; Zhang, Q. *ZigBee® Network Protocols and Applications*; Auerbach Publications: Boca Raton, FL, USA, 2016.
38. Yassein, M.B.; Mardini, W.; Khalil, A. Smart homes automation using Z-wave protocol. In Proceedings of the International Conference on Engineering & MIS (ICEMIS), Agadir, Morocco, 22–24 September 2016.
39. Vagdevi, P.; Nagaraj, D.; Prasad, G.V. Home: IOT based home automation using NFC. In Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, Palladam, India, 10–11 February 2017.
40. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
41. Lin, J.; Yu, W.; Zhang, N.; Yang, X.; Zhang, H.; Zhao, W. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet Things J.* **2017**, *4*, 1125–1142. [CrossRef]
42. Pattar, S.; Buyya, R.; Venugopal, K.; Iyengar, S.; Patnaik, L. Searching for the IoT Resources: Fundamentals, Requirements, Comprehensive Review and Future Directions. *IEEE Commun. Surv. Tutor.* **2018**. [CrossRef]
43. Alaba, F.A.; Othman, M.; Hashem, I.A.T.; Alotaibi, F. Internet of Things security: A survey. *J. Netw. Comput. Appl.* **2017**, *88*, 10–28. [CrossRef]
44. Tian, H.; Nan, F.; Chang, C.C.; Huang, Y.; Lu, J.; Du, Y. Privacy-preserving public auditing for secure data storage in fog-to-cloud computing. *J. Netw. Comput. Appl.* **2019**, *127*, 59–69. [CrossRef]
45. Zhang, Y.; Deng, R.H.; Han, G.; Zheng, D. Secure smart health with privacy-aware aggregate authentication and access control in Internet of Things. *J. Netw. Comput. Appl.* **2018**, *123*, 89–100. [CrossRef]
46. Barki, A.; Bouabdallah, A.; Gharout, S.; Traoré, J. M2M security: Challenges and solutions. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1241–1254. [CrossRef]
47. Ni, J.; Zhang, K.; Lin, X.; Shen, X.S. Securing fog computing for internet of things applications: Challenges and solutions. *IEEE Commun. Surv. Tutor.* **2017**, *20*, 601–628. [CrossRef]
48. Stellios, I.; Kotzanikolaou, P.; Psarakis, M.; Alcaraz, C.; Lopez, J. A Survey of IoT-enabled Cyberattacks: Assessing Attack Paths to Critical Infrastructures and Services. *IEEE Commun. Surv. Tutor.* **2018**. [CrossRef]
49. Benkhelifa, E.; Welsh, T.; Hamouda, W. A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Towards Universal and Resilient Systems. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3496–3509. [CrossRef]
50. Mohanta, B.K.; Jena, D.; Satapathy, U.; Patnaik, S. Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology. *Internet Things* **2020**, *11*, 100227. [CrossRef]
51. Al-Garadi, M.A.; Mohamed, A.; Al-Ali, A.K.; Du, X.; Ali, I.; Guizani, M. A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1646–1685. [CrossRef]
52. Stoyanova, M.; Nikoloudakis, Y.; Panagiotakis, S.; Pallis, E.; Markakis, E.K. A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1191–1221. [CrossRef]
53. Lounis, K.; Zulkernine, M. Attacks and defenses in short-range wireless technologies for IoT. *IEEE Access* **2020**, *8*, 88892–88932. [CrossRef]
54. Sharma, V.; You, I.; Andersson, K.; Palmieri, F.; Rehmani, M.H.; Lim, J. Security, privacy and trust for smart mobile-Internet of Things (M-IoT): A survey. *IEEE Access* **2020**, *8*, 167123–167163. [CrossRef]
55. Sha, K.; Yang, T.A.; Wei, W.; Davari, S. A survey of edge computing-based designs for iot security. *Digit. Commun. Netw.* **2020**, *6*, 195–202. [CrossRef]
56. Tahsien, S.M.; Karimipour, H.; Spachos, P. Machine learning based solutions for security of Internet of Things (IoT): A survey. *J. Netw. Comput. Appl.* **2020**, *161*, 102630. [CrossRef]
57. Hassija, V.; Chamola, V.; Saxena, V.; Jain, D.; Goyal, P.; Sikdar, B. A survey on IoT security: Application areas, security threats, and solution architectures. *IEEE Access* **2019**, *7*, 82721–82743. [CrossRef]
58. Neshenko, N.; Bou-Harb, E.; Crichigno, J.; Kaddoum, G.; Ghani, N. Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2702–2733. [CrossRef]
59. Mrabet, H.; Belguith, S.; Alhomoud, A.; Jemai, A. A survey of IoT security based on a layered architecture of sensing and data analysis. *Sensors* **2020**, *20*, 3625. [CrossRef]

60. Ahmed, A.; Abdullah, S.; Bukhsh, M.; Ahmad, I.; Mushtaq, Z. An energy-efficient data aggregation mechanism for IoT secured by blockchain. *IEEE Access* **2022**, *10*, 11404–11419. [CrossRef]
61. Hewa, T.; Braeken, A.; Liyanage, M.; Ylianttila, M. Fog computing and blockchain-based security service architecture for 5G industrial IoT-enabled cloud manufacturing. *IEEE Trans. Ind. Inform.* **2022**, *18*, 7174–7185. [CrossRef]
62. Javanmardi, S.; Shojafar, M.; Mohammadi, R.; Persico, V.; Pescapè, A. S-FoS: A secure workflow scheduling approach for performance optimization in SDN-based IoT-Fog networks. *J. Inf. Secur. Appl.* **2023**, *72*, 103404. [CrossRef]
63. Javanmardi, S.; Shojafar, M.; Mohammadi, R.; Nazari, A.; Persico, V.; Pescapè, A. FUPE: A security driven task scheduling approach for SDN-based IoT–Fog networks. *J. Inf. Secur. Appl.* **2021**, *60*, 102853. [CrossRef]
64. Javanmardi, S.; Shojafar, M.; Mohammadi, R.; Alazab, M.; Caruso, A.M. An SDN perspective IoT-Fog security: A survey. *Comput. Netw.* **2023**, *229*, 109732. [CrossRef]
65. Meneghello, F.; Calore, M.; Zucchetto, D.; Polese, M.; Zanella, A. IoT: Internet of threats? A survey of practical security vulnerabilities in real IoT devices. *IEEE Internet Things J.* **2019**, *6*, 8182–8201. [CrossRef]
66. Desai, P.; Sheth, A.; Anantharam, P. Semantic gateway as a service architecture for IoT interoperability. In Proceedings of the 2015 IEEE International Conference on Mobile Services, New York, NY, USA, 27 June–2 July 2015.
67. Datta, S.K.; Bonnet, C.; Nikaein, N. An IoT gateway centric architecture to provide novel m2m services. In Proceedings of the 2014 IEEE World Forum on Internet of Things (WF-IoT), Seoul, Republic of Korea, 6–8 March 2014.
68. 6LoWPAN. IPv6 over Low-Power Wireless Personal Area Networks. RFC4919. 2016. Available online: https://www.rfc-editor.org/rfc/rfc4919 (accessed on 22 January 2024).
69. ZigBee. ZigBee Specification. 2015. Available online: https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf (accessed on 22 January 2024).
70. Lin, Z.M.; Chang, C.H.; Chou, N.K.; Lin, Y.H. Bluetooth Low Energy (BLE) based blood pressure monitoring system. In Proceedings of the International Conference on Intelligent Green Building and Smart Grid (IGBSG), Taipei, Taiwan, 23–25 April 2014.
71. Shelby, Z.; Hartke, K.; Bormann, C. The Constrained Application Protocol (CoAP). RFC IETF. 2014. Available online: https://www.rfc-editor.org/rfc/rfc7252 (accessed on 22 January 2024).
72. Gaddour, O.; Koubâa, A. RPL in a nutshell: A survey. *Comput. Netw.* **2012**, *56*, 3163–3178. [CrossRef]
73. Verma, A.; Bhardwaj, N. A review on routing information protocol (RIP) and open shortest path first (OSPF) routing protocol. *Int. J. Future Gener. Commun. Netw.* **2016**, *9*, 161–170. [CrossRef]
74. Covington, M.; Carskadden, R. Threat implications of the Internet of Things. In Proceedings of the IEEE International Conference on Communications (ICC), Budapest, Hungary, 9–13 June 2013.
75. Gruschka, N.; Jensen, M. Attack surfaces: A taxonomy for attacks on cloud service. In Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, Miami, FL, USA, 5–10 July 2010.
76. OWASP. Open Web Application Security Project for Internet of Things. 2015. Available online: https://owasp.org/www-project-internet-of-things/ (accessed on 22 January 2024).
77. Lake, D.; Milito, R.; Morrow, M.; Vargheese, R. Internet of Things: Architectural Framework for eHealth Security. *J. ICT Stand.* **2014**, *1*, 301–328. [CrossRef]
78. Sanchez, J.L.C.; Bernabe, J.B.; Skarmeta, A.F. Towards privacy preserving data provenance for the Internet of Things. In Proceedings of the 4th World Forum on Internet of Things (WF-IoT), Singapore, 5–8 February 2018.
79. Medwed, M. Iot security challenges and ways forward. In *International Workshop on Trustworthy Embedded Devices*; ACM: New York, NY, USA, 2016; p. 55.
80. Boulogeorgos, A.A.A.; Diamantoulakis, P.D.; Karagiannidis, G.K. Low power wide area networks (lpwans) for internet of things (iot) applications: Research challenges and future trends. *arXiv* **2016**, arXiv:1611.07449.
81. Rathi, N.; Ghosh, S.; Iyengar, A.; Naeimi, H. Data privacy in non-volatile cache: Challenges, attack models and solutions. In Proceedings of the 21st Asia and South Pacific Design Automation Conference (ASP-DAC), Macao, China, 25–28 January 2016.
82. Kubler, S.; Främling, K.; Buda, A. A standardized approach to deal with firewall and mobility policies in the IoT. *Pervasive Mob. Comput.* **2015**, *20*, 100–114. [CrossRef]
83. Qian, L.; Zhu, Z.; Hu, J.; Liu, S. Research of SQL injection attack and prevention technology. In Proceedings of the Detection and Information Fusion (ICEDIF), 2015 International Conference on Estimation, Harbin, China, 10–11 January 2015.
84. Aggarwal, S.; Houshmand, S.; Weir, M. New Technologies in Password Cracking Techniques. In *Cyber Security: Power and Technology*; Spronger: Berlin/Heidelberg, Germany, 2018; pp. 179–198.
85. Gupta, B.; Gupta, S.; Gangwar, S.; Kumar, M.; Meena, P. Cross-site scripting (XSS) abuse and defense: Exploitation on several testing bed environments and its defense. *J. Inf. Priv. Secur.* **2015**, *11*, 118–136. [CrossRef]
86. Neamtiu, I.; Dumitraş, T. Cloud software upgrades: Challenges and opportunities. In Proceedings of the International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), Williamsburg, VA, USA, 26 September 2011.
87. Stevens, M.; Bursztein, E.; Karpman, P.; Albertini, A.; Markov, Y.; Bianco, A.P.; Baisse, C. SHA1 Collision. Cryptology ePrint Archive, Paper 2017/190. Available online: https://eprint.iacr.org/2017/190 (accessed on 22 January 2024).
88. Yuan, M.; Li, Y.; Li, Z. Hijacking Your Routers via Control-Hijacking URLs in Embedded Devices with Web Interfaces. In Proceedings of the International Conference on Information and Communications Security, Beijing, China, 6–8 December 2017.

89. Antonakakis, M.; April, T.; Bailey, M.; Bernhard, M.; Bursztein, E.; Cochran, J.; Durumeric, Z.; Halderman, J.A.; Invernizzi, L.; Kallitsis, M.; et al. Understanding the mirai botnet. In Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, USA, 16–18 August 2017; pp. 1093–1110.
90. Ko, J.; Jeong, J.; Park, J.; Jun, J.A.; Gnawali, O.; Paek, J. DualMOP-RPL: Supporting multiple modes of downward routing in a single RPL network. *ACM Trans. Sens. Netw.* **2015**, *11*, 39. [CrossRef]
91. Open Web Application Security Project (OWASP). The Heartbleed Bug. Available online: https://owasp.org/www-community/vulnerabilities/Heartbleed_Bug (accessed on 22 January 2024).
92. Palacharla, S.; Chandan, M.; GnanaSuryaTeja, K.; Varshitha, G. Wormhole Attack: A Major Security Concern in Internet of Things (Iot). *Int. J. Eng. Technol.* **2018**, *7*, 147–150. [CrossRef]
93. Liu, Y.; Ma, M.; Liu, X.; Xiong, N.; Liu, A.; Zhu, Y. Design and Analysis of Probing Route to Defense Sink-hole Attacks for Internet of Things Security. *IEEE Trans. Netw. Sci. Eng.* **2018**, *7*, 356–372. [CrossRef]
94. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and other botnets. *Computer* **2017**, *50*, 80–84. [CrossRef]
95. Alberca, C.; Pastrana, S.; Suarez-Tangil, G.; Palmieri, P. Security analysis and exploitation of arduino devices in the internet of things. In Proceedings of the ACM International Conference on Computing Frontiers, Como, Italy, 16–19 May 2016.
96. Pongle, P.; Chavan, G. A survey: Attacks on RPL and 6LoWPAN in IoT. In Proceedings of the Pervasive Computing (ICPC), 2015 International Conference on Pervasive Computing (ICPC), Pune, India, 8–10 January 2015.
97. Salameh, H.B.; Almajali, S.; Ayyash, M.; Elgala, H. Securing delay-sensitive cognitive radio IoT communications under reactive jamming attacks: Spectrum assignment perspective. In Proceedings of the 2018 Fifth International Conference on Software Defined Systems (SDS), Barcelona, Spain, 23–26 April 2018.
98. Chiew, K.L.; Yong, K.S.C.; Tan, C.L. A survey of phishing attacks: Their types, vectors and technical approaches. *Expert Syst. Appl.* **2018**, *106*, 1–20. [CrossRef]
99. Lyon, G. Nmap: A Network Mapper. Available online: https://nmap.org/#:~:text=Nmap%20(%22Network%20Mapper%22),monitoring%20host%20or%20service%20uptime (accessed on 22 January 2024).
100. Krupp, J.; Backes, M.; Rossow, C. Identifying the scan and attack infrastructures behind amplification DDoS attacks. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016.
101. THC-Hydra. Hydra: A Password Guessing Tool. Available online: https://github.com/vanhauser-thc/thc-hydra (accessed on 22 January 2024).
102. Chen, C.K.; Zhang, Z.K.; Lee, S.H.; Shieh, S. Penetration testing in the iot age. *Computer* **2018**, *51*, 82–85. [CrossRef]
103. Visoottiviseth, V.; Akarasiriwong, P.; Chaiyasart, S.; Chotivatunyu, S. PENTOS: Penetration testing tool for Internet of Thing devices. In Proceedings of the TENCON 2017—2017 IEEE Region 10 Conference, Penang, Malaysia, 5–8 November 2017.
104. Chothia, T.; de Ruiter, J. Learning from others mistakes: Penetration testing iot devices in the classroom. In Proceedings of the 2016 USENIX Workshop on Advances in Security Education (ASE 16), Austin, TX, USA, 9 August 2016.
105. Forbes. Roundup of Internet of Things Forecasts. 2017. Available online: https://goo.gl/iVf5uz (accessed on 22 January 2024).
106. Hu, W.; Tan, H.; Corke, P.; Shih, W.C.; Jha, S. Toward trusted wireless sensor networks. *ACM Trans. Sens. Netw.* **2010**, *7*, 1–25. [CrossRef]
107. Smith, S.W.; Weingart, S. Building a high-performance, programmable secure coprocessor. *Comput. Netw.* **1999**, *31*, 831–860. [CrossRef]
108. Costan, V.; Lebedev, I.; Devadas, S. Secure processors part I: Background, taxonomy for secure enclaves and Intel SGX architecture. *Found. Trends® Electron. Des. Autom.* **2017**, *11*, 1–248. [CrossRef]
109. Pinto, S.; Gomes, T.; Pereira, J.; Cabral, J.; Tavares, A. IIoTEED: An enhanced, trusted execution environment for industrial IoT edge devices. *IEEE Internet Comput.* **2017**, *21*, 40–47. [CrossRef]
110. Hadi, N.; Jim, R. Employ a Secure Flavor of Linux. 2007. Available online: https://www.embedded.com/employ-a-secure-flavor-of-linux/ (accessed on 22 January 2024).
111. Hennessy, A.; Zheng, Y.; Bhunia, S. JTAG-based robust PCB authentication for protection against counterfeiting attacks. In Proceedings of the 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), Macao, China, 25–28 January 2016.
112. Das, A.; Da Rolt, J.; Ghosh, S.; Seys, S.; Dupuis, S.; Di Natale, G.; Flottes, M.L.; Rouzeyre, B.; Verbauwhede, I. Secure JTAG implementation using schnorr protocol. *J. Electron. Test.* **2013**, *29*, 193–209. [CrossRef]
113. Rosenfeld, K.; Karri, R. Attacks and Defenses for JTAG. *Des. Test Comput.* **2010**. [CrossRef]
114. Zhao, L.; Misoczki, R.; Ghosh, S.; Sastry, M.R. Root of Trust (Rot) Application for Internet of Things (IoT) Devices. U.S. Patent App. 15/278,658, 29 March 2018.
115. Belenky, Y.; Sumner, R. Prevention of Playback Attacks Using OTP Memory. U.S. Patent 9,009,492, 19 June 2015.
116. Fifield, J.A.; Pomichter G.P., Jr.; Zimmerman, J.S. Protection of One-Time Programmable (OTP) Memory. U.S. Patent 8,990,478, 8 May 2015.
117. Moskowitz, R.; Heer, T.; Jokela, P.; Henderson, T. Host Identity Protocol Version 2 (HIPv2). RFC, IETF. 2015. Available online: https://www.rfc-editor.org/rfc/rfc7401.html (accessed on 22 January 2024).
118. Saied, Y.B.; Olivereau, A. D-HIP: A distributed key exchange scheme for HIP-based Internet of Things. In Proceedings of the WoWMoM, New York, NY, USA, 24–25 October 2012.
119. Ben Saied, Y.; Olivereau, A. HIP Tiny Exchange (TEX): A distributed key exchange scheme for HIP-based Internet of Things. In Proceedings of the CNS, Hammamet, Tunisia, 29 March–1 April 2012.

120. Hummen, R.; Hiller, J.; Henze, M.; Wehrle, K. Slimfit—A HIP DEX compression layer for the IP-based Internet of Things. In Proceedings of the 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Lyon, France, 7–9 October 2013.
121. Hummen, R.; Moskowitz, R. HIP Diet EXchange (DEX). RFC, IETF. 2014. Available online: http://www.watersprings.org/pub/id/draft-ietf-hip-dex-18.html (accessed on 22 January 2024).
122. Garcia-Morchon, O.; Keoh, S.L.; Kumar, S.; Moreno-Sanchez, P.; Vidal-Meca, F.; Ziegeldorf, J.H. Securing the IP-based internet of things with HIP and DTLS. In Proceedings of the WiSec: Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, Budapest Hungary, 17–19 April 2013.
123. Blundo, C.; De Santis, A.; Herzberg, A.; Kutten, S.; Vaccaro, U.; Yung, M. Perfectly-secure key distribution for dynamic conferences. In Advances in Cryptology–CRYPTO. Available online: https://link.springer.com/chapter/10.1007/3-540-48071-4_33 (accessed on 22 January 2024).
124. Chen, L. *Recommendation for Key Derivation Using Pseudorandom Functions*; NIST Special Publication: Gaithersburg, MD, USA, 2008. Available online: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-108.pdf (accessed on 22 January 2024).
125. Alexander, R.; Tsao, T. Adapted Multimedia Internet KEYing (AMIKEY): An extension of Multimedia Internet KEYing (MIKEY) Nethods for Generic LLN Environments. RFC, IETF 2012. Available online: https://www.ietf.org/archive/id/draft-alexander-roll-mikey-lln-key-mgmt-02.html (accessed on 22 January 2024).
126. Heer, T. LHIP: Lightweight Authentication Extension for HIP. RFC, IETF 2007. Available online: https://datatracker.ietf.org/doc/draft-heer-hip-lhip/ (accessed on 22 January 2024).
127. Hossain, M.; Hasan, R. P-hip: A lightweight and privacy-aware host identity protocol for internet of things. *IEEE Internet Things J.* **2020**, *8*, 555–571. [CrossRef]
128. Kothmayr, T.; Schmitt, C.; Hu, W.; Brunig, M.; Carle, G. A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication. In Proceedings of the 37th Annual IEEE Conference on Local Computer Networks-Workshops, Clearwater, FL, USA, 22–25 October 2012.
129. Eronen, P.; Tschofenig, H. Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). RFC, IETF. 2005. Available online: https://www.rfc-editor.org/rfc/rfc4279.html (accessed on 22 January 2024).
130. Hui, J.; Thubert, P. Compression Format for IPv6 Datagrams over IEEE 802.15. 4-Based Networks. IETF, RFC. 2011. Available online: https://www.rfc-editor.org/rfc/rfc6282 (accessed on 22 January 2024).
131. Park, J.; Kang, N. Lightweight secure communication for CoAP-enabled Internet of Things using delegated DTLS handshake. In Proceedings of the 2014 International Conference on Information and Communication Technology Convergence (ICTC), Busan, Republic of Korea, 22–24 October 2014.
132. Hummen, R.; Ziegeldorf, J.H.; Shafagh, H.; Raza, S.; Wehrle, K. Towards viable certificate-based authentication for the Internet of Things. In Proceedings of the 2nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy, Budapest, Hungary, 19 April 2013.
133. Hummen, R.; Shafagh, H.; Gilger, J. Extended DTLS Session Resumption for Constrained Network Environments. RFC, IETF. 2013. Available online: https://datatracker.ietf.org/doc/html/draft-hummen-dtls-extended-session-resumption-01 (accessed on 22 January 2024).
134. Liu, J.; Xiao, Y.; Chen, C.P. Authentication and access control in the Internet of Things. In Proceedings of the 2012 32nd International Conference on Distributed Computing Systems Workshops, Macau, China, 18–21 June 2012.
135. Zhang, G.; Tian, J. An extended role based access control model for the Internet of Things. In Proceedings of the 2010 International Conference on Information, Networking and Automation (ICINA), Kunming, China, 18–19 October 2010.
136. Gusmeroli, S.; Piccione, S.; Rotondi, D. A capability-based security approach to manage access control in the Internet of Things. *Math. Comput. Model.* **2013**, *58*, 1189–1205. [CrossRef]
137. Seitz, L.; Selander, G.; Gehrmann, C. Authorization framework for the Internet-of-Things. In Proceedings of the 2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Madrid, Spain, 4–7 June 2013.
138. Pereira, P.P.; Eliasson, J.; Delsing, J. An authentication and access control framework for CoAP-based Internet of Things. In Proceedings of the IECON 2014—40th Annual Conference of the IEEE Industrial Electronics Society, Dallas, TX, USA, 29 October–1 November 2014.
139. Hernández-Ramos, J.L.; Jara, A.J.; Marın, L.; Skarmeta, A.F. Distributed capability-based access control for the Internet of Things. *J. Internet Serv. Inf. Secur.* **2013**, *3*, 1–16.
140. Mahalle, P.N.; Anggorojati, B.; Prasad, N.R.; Prasad, R. Identity authentication and capability based access control (IACAC) for the Internet of Things. *J. Cyber Secur. Mobil.* **2013**, *1*, 309–348. [CrossRef]
141. Hummen, R.; Shafagh, H.; Raza, S.; Voig, T.; Wehrle, K. Delegation-based authentication and authorization for the IP-based Internet of Things. In Proceedings of the 2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), Singapore, 30 June–3 July 2014.
142. Anggorojati, B.; Mahalle, P.N.; Prasad, N.R.; Prasad, R. Capability-based access control delegation model on the federated IoT network. In Proceedings of the 15th International Symposium on Wireless Personal Multimedia Communications, Taipei, Taiwan, 24–27 September 2012.

143. Cirani, S.; Picone, M.; Gonizzi, P.; Veltri, L.; Ferrari, G. IoT-OAS: An OAuth-based authorization service architecture for secure services in IoT scenarios. *J. Sens.* **2015**, *15*, 1224–1234. [CrossRef]
144. Gerdes, S.; Bergmann, O.; Bormann, C. Delegated CoAP Authentication and Authorization Framework (DCAF). ETF Internet Draft, 2014. Available online: https://datatracker.ietf.org/doc/draft-gerdes-core-dcaf-authorize/01/ (accessed on 22 January 2024).
145. Moratelli, C.; Johann, S.; Neves, M.; Hessel, F. Embedded virtualization for the design of secure IoT applications. In Proceedings of the 27th International Symposium on Rapid System Prototyping: Shortening the Path from Specification to Prototype, Pittsburgh, PA, USA, 1–7 October 2016.
146. Lu, W.; Wang, R.; Zeng, C.; Liu, C.; Wang, X. A General Fault Injection Method Based on JTAG. In Proceedings of the 2018 Prognostics and System Health Management Conference (PHM-Chongqing), Chongqing, China, 26–28 October 2018.
147. Shelby, Z.; Bormann, C. *6LoWPAN: The Wireless Embedded Internet*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
148. Butun, I.; Sankar, R. A brief survey of access control in Wireless Sensor Networks. In Proceedings of the 2011 IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2011.
149. Wang, H.; Sheng, B.; Tan, C.C.; Li, Q. Comparing symmetric-key and public-key based security schemes in sensor networks: A case study of user access control. In Proceedings of the International Conference on Distributed Computing Systems, Beijing, China, 17–20 June 2008.
150. NIST. *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2018.
151. Amin, F.; Jahangir, A.; Rasifard, H. Analysis of public-key cryptography for wireless sensor networks security. *Int. J. Comput. Inf. Eng.* **2008**, *2*, 1448–1453.
152. Hu, W.; Corke, P.; Shih, W.C.; Overs, L. Secfleck: A public key technology platform for wireless sensor networks. In *Wireless Sensor Networks*; Springer: Berlin/Heidelberg, Germany, 2009.
153. Kocabas, O.; Savas, E.; Großschädl, J. Enhancing an Embedded Processor Core with a Cryptographic Unit for Performance and Security. In Proceedings of the 2008 International Conference on Reconfigurable Computing and FPGAs, Cancun, Mexico, 3–5 December 2008.
154. Liu, A.; Ning, P. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In Proceedings of the 2008 International Conference on Information Processing in Sensor Networks (ipsn 2008), St. Louis, MO, USA, 22–24 April 2008.
155. Guicheng, S.; Zhen, Y. Application of elliptic curve cryptography in node authentication of Internet of Things. In Proceedings of the 2013 Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Beijing, China, 16–18 October 2013.
156. Gupta, V.; Wurm, M.; Zhu, Y.; Millard, M.; Fung, S.; Gura, N.; Eberle, H.; Shantz, S.C. Sizzle: A standards-based end-to-end security architecture for the embedded Internet. *Pervasive Mob. Comput.* **2005**, *1*, 425–445. [CrossRef]
157. Bohan, Z.; Xu, W.; Kaili, Z.; Xueyuan, Z. Encryption Node Design in Internet of Things Based on Fingerprint Features and CC253. In Proceedings of the 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, Beijing, China, 20–23 August 2013.
158. Ravi, S.; Raghunathan, A.; Potlapally, N.; Sankaradass, M. System design methodologies for a wireless security processing platform. In Proceedings of the 39th Annual Design Automation Conference, New Orleans, LA, USA, 10–14 June 2002.
159. Wander, A.S.; Gura, N.; Eberle, H. Energy Analysis of Public–key Cryptography on Small Wireless Devices. In Proceedings of the 3rd IEEE Intl Conference on Pervasive Computing and Communications, Kauai, HI, USA, 8–12 March 2005.
160. Kanuparthi, A.; Karri, R.; Addepalli, S. Hardware and embedded security in the context of Internet of Things. In Proceedings of the 2013 ACM Workshop on Security, Privacy & Dependability for Cyber Vehicles, Berlin, Germany, 4 November 2013.
161. Levä, T.; Mazhelis, O.; Suomi, H. Comparing the cost-efficiency of CoAP and HTTP in Web of Things applications. *Decis. Support Syst.* **2014**, *63*, 23–38. [CrossRef]
162. Ferraiolo, D.; Cugini, J.; Kuhn, D.R. Role-based access control (RBAC): Features and motivations. In Proceedings of the ACSAC; IEEE: Piscataway, NJ, USA, 11–15 December 1995.
163. Pesonen, L.I.; Eyers, D.M.; Bacon, J. A capability-based access control architecture for multi-domain publish/subscribe systems. In Proceedings of the International Symposium on Applications and the Internet (SAINT'06), Phoenix, AZ, USA, 23–27 January 2006.
164. Recordon, D.; Reed, D. OpenID 2.0: A platform for user-centric identity management. In Proceedings of the Second ACM Workshop on Digital Identity Management, Alexandria, VA, USA, 3 November 2006.
165. Godik, S.; Moses, T.; Anderson, A.; Parducci, B.; Adams, C.; Flinn, D.; Brose, G.; Lockhart, H.; Beznosov, K.; Kudo, M.; et al. Extensible access control markup language (XACMl) version 2.0. *Oasis Stand.* **2005**. Available online: https://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf (accessed on 22 January 2024).
166. Crockford, D. The Application/Json Media Type for Javascript Object Notation (JSON). RFC, IETF. 2006. Available online: https://www.rfc-editor.org/rfc/rfc4627.html (accessed on 22 January 2024).
167. Zhu, L.; Hartman, S.; Jaganathan, K. The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2. RFC, IETF. 2005. Available online: https://www.rfc-editor.org/rfc/rfc4121 (accessed on 22 January 2024).
168. Willens, S.; Rubens, A.C.; Rigney, C.; Simpson, W.A. Remote Authentication Dial in User Service (RADIUS). RFC, NWG. 2000. Available online: https://www.rfc-editor.org/rfc/rfc2865.html (accessed on 22 January 2024).

169. Hardt, D. The OAuth 2.0 Authorization Framework. RFC, IETF. 2012. Available online: https://datatracker.ietf.org/doc/html/rfc6749 (accessed on 22 January 2024).
170. Kayas, G.; Hossain, M.; Payton, J.; Islam, S.R. SUPnP: Secure Access and Service Registration for UPnP-Enabled Internet of Things. *IEEE Internet Things J.* **2021**, *8*, 11561–11580. [CrossRef]
171. Asghar, M.H.; Negi, A.; Mohammadzadeh, N. Principle application and vision in Internet of Things (IoT). In Proceedings of the International Conference on Computing, Communication & Automation, Greater Noida, India, 15–16 May 2015.
172. Anderson, C.; Hübener, I.; Seipp, A.K.; Ohly, S.; David, K.; Pejovic, V. A survey of attention management systems in ubiquitous computing environments. *Proc. Acm Interact. Mob. Wearable Ubiquitous Technol.* **2018**, *2*, 58. [CrossRef]
173. Ling, C.H.; Lee, C.C.; Yang, C.C.; Hwang, M.S. A Secure and Efficient One-time Password Authentication Scheme for WSN. *Int. J. Netw. Secur.* **2017**, *19*, 177–181.
174. Chuang, I.H.; Guo, B.J.; Tsai, J.S.; Kuo, Y.H. Multi-graph Zero-knowledge-based authentication system in Internet of Things. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017.
175. Bernal Bernabe, J.; Hernandez-Ramos, J.L.; Skarmeta Gomez, A.F. Holistic privacy-preserving identity management system for the internet of things. *Mob. Inf. Syst.* **2017**, *2017*, 20. [CrossRef]
176. amsdell, B.; Turner, S. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC, IETF. 2010. Available online: https://www.rfc-editor.org/rfc/rfc5751.html (accessed on 22 January 2024).
177. Baugher, M.; McGrew, D.; Naslund, M.; Carrara, E.; Norrman, K. The Secure Real-Time Transport Protocol (SRTP). RFC, IETF. 2004. Available online: https://www.rfc-editor.org/rfc/rfc3711.html (accessed on 22 January 2024).
178. Levis, P.; Madden, S.; Polastre, J.; Szewczyk, R.; Whitehouse, K.; Woo, A.; Gay, D.; Hill, J.; Welsh, M.; Brewer, E.; et al. Tinyos: An operating system for sensor networks. In Ambient Intelligence. Available online: https://www.researchgate.net/publication/228639896_TinyOS_An_Operating_System_for_Sensor_Networks (accessed on 22 January 2024).
179. Lu, G.; Nam, Y.J.; Du, D.H. BloomStore: Bloom-filter based memory-efficient key-value store for indexing of data deduplication on flash. In Proceedings of the IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST), Pacific Grove, CA, USA, 16–20 April 2012.
180. Herder, C.; Ren, L.; van Dijk, M.; Yu, M.D.; Devadas, S. Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions. *IEEE Trans. Dependable Secur. Comput.* **2017**, *14*, 65–82. [CrossRef]
181. Dong, P.; Wang, W.; Shi, X.; Qin, T. Lightweight key management for group communication in body area networks through physical unclonable functions. In Proceedings of the Second IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies, Philadelphia, PA, USA, 17–19 July 2017.
182. Valsesia, D.; Coluccia, G.; Bianchi, T.; Magli, E. User Authentication via PRNU-Based Physical Unclonable Functions. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 1941–1956. [CrossRef]
183. Hossain, M.; Noor, S.; Hasan, R. HSC-IoT: A Hardware and Software Co-Verification based Authentication Scheme for Internet of Things. In Proceedings of the 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), San Francisco, CA, USA, 6–8 April 2017.
184. Suárez-Albela, M.; Fernández-Caramés, T.M.; Fraga-Lamas, P.; Castedo, L. A practical performance comparison of ECC and RSA for resource-constrained IoT devices. In Proceedings of the 2018 Global Internet of Things Summit (GIoTS), Bilbao, Spain, 4–7 June 2018.
185. Kornaros, G. Hardware-assisted machine learning in resource-constrained IoT environments for security: Review and future prospective. *IEEE Access* **2022**, *10*, 58603–58622. [CrossRef]
186. Hossain, M.; Kayas, G.; Karim, Y.; Hasan, R.; Payton, J.; Islam, S.R. CATComp: A Compression-Aware Authorization Protocol for Resource-Efficient Communications in IoT Networks. *IEEE Internet Things J.* **2021**, *9*, 1667–1682. [CrossRef]
187. Chandran, P. Secure and Dynamic Memory Management Architecture for Virtualization Technologies in IoT Devices. *Future Internet* **2018**, *10*, 119.
188. Khan, M.N.; Rao, A.; Camtepe, S. Lightweight cryptographic protocols for IoT-constrained devices: A survey. *IEEE Internet Things J.* **2020**, *8*, 4132–4156. [CrossRef]
189. Zahed, M.I.A.; Ahmad, I.; Habibi, D.; Phung, Q.V. Green and secure computation offloading for cache-enabled IoT networks. *IEEE Access* **2020**, *8*, 63840–63855. [CrossRef]
190. Chang, C.C.; Lee, W.K.; Liu, Y.; Goi, B.M.; Phan, R.C.W. Signature gateway: Offloading signature generation to IoT gateway accelerated by GPU. *IEEE Internet Things J.* **2018**, *6*, 4448–4461. [CrossRef]
191. El Jaouhari, S.; Bouvet, E. Secure firmware Over-The-Air updates for IoT: Survey, challenges, and discussions. *Internet Things* **2022**, *18*, 100508. [CrossRef]
192. Arakadakis, K.; Charalampidis, P.; Makrogiannakis, A.; Fragkiadakis, A. Firmware over-the-air programming techniques for IoT networks-A survey. *ACM Comput. Surv. (Csur)* **2021**, *54*, 1–36. [CrossRef]
193. Suricata. A High Performance, Open Source Network Analysis and Threat Detection Software. Available online: https://suricata.io/ (accessed on 22 January 2024).
194. Beale, J.; Baker, A.R.; Esler, J. Snort: IDS and IPS Toolkit. Available online: https://www.snort.org/ (accessed on 22 January 2024).
195. KitPloit. Moloch: An Open Source, Large Scale, Full Packet Capturing, Indexing, and Database System. Available online: https://www.kitploit.com/2018/04/moloch-open-source-large-scale-full.html?m=0 (accessed on 22 January 2024).

196. Zawoad, S.; Hasan, R. FAIoT: Towards building a forensics aware eco system for the Internet of Things. In Proceedings of the 2015 IEEE International Conference on Services Computing, New York, NY, USA, 27 June–2 July 2015.
197. Widup, S. *Computer Forensics and Digital Investigation with EnCase Forensic v7*; McGraw-Hill Education Group: New York, NY, USA, 2014.
198. Zawoad, S.; Dutta, A.K.; Hasan, R. Towards building forensics enabled cloud through secure logging-as-a-service. *IEEE Trans. Dependable Secur. Comput.* **2016**, *13*, 148–162. [CrossRef]
199. Khan, S.; Gani, A.; Wahab, A.W.A.; Shiraz, M.; Ahmad, I. Network forensics: Review, taxonomy, and open challenges. *J. Netw. Comput. Appl.* **2016**, *66*, 214–235. [CrossRef]
200. Meffert, C.; Clark, D.; Baggili, I.; Breitinger, F. Forensic State Acquisition from Internet of Things (FSAIoT): A general framework and practical approach for IoT forensics through IoT device state acquisition. In Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, 29 August–1 September 2017.
201. Zia, T.; Liu, P.; Han, W. Application-Specific Digital Forensics Investigative Model in Internet of Things (IoT). In Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, 29 August–1 September 2017.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.