Alma Mater Studiorum - Università di Bologna

Dottorato di Ricerca in

COMPUTER SCIENCE AND ENGINEERING

Ciclo XXXVI

Settore concorsuale:
09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore scientifico disciplinare:
ING-INF/05 SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Adaptable Security in the Cloud-to-Thing Continuum

Presentata da: Carlo Mazzocca

Coordinatore Dottorato:
Prof.ssa Ilaria Bartolini

Supervisore:
Prof.ssa Rebecca Montanari

Esame finale anno 2024

Abstract

Recent technological advancements have played a key role in seamlessly integrating cloud, edge, and Internet of Things (IoT) technologies, giving rise to the Cloud-to-Thing Continuum paradigm. This cloud model connects many heterogeneous resources that generate a large amount of data and collaborate to deliver next-generation services. While it has the potential to reshape several application domains, the number of connected entities remarkably broadens the security attack surface. One of the main problems is the lack of security measures to adapt to the dynamic and evolving conditions of the Cloud-To-Thing Continuum.

To address this challenge, this dissertation proposes novel adaptable security mechanisms. Adaptable security is the capability of security controls, systems, and protocols to dynamically adjust to changing conditions and scenarios. However, since the design and development of novel security mechanisms can be explored from different perspectives and levels, we place our attention on threat modeling and access control.

The contributions of the thesis can be summarized as follows. First, we introduce a model-based methodology that secures the design of edge and cyber-physical systems. This solution identifies threats, security controls, and moving target defense techniques based on system features.

Then, we focus on access control management. Since access control policies are subject to modifications, we evaluate how they can be efficiently shared among distributed areas, highlighting the effectiveness of distributed ledger technologies. Furthermore, we propose a risk-based authorization middleware, adjusting permissions based on real-time data, and a federated learning framework that enhances trustworthiness by weighting each client's contributions according to the quality of their partial models.

Finally, since authorization revocation is another critical concern, we present an efficient revocation scheme for verifiable credentials in IoT networks, featuring decentralization, demanding minimum storage and computing capabilities.

All the mechanisms have been evaluated in different conditions, proving their adaptability to the Cloud-to-Thing Continuum landscape.

Contents

List of Tables

List of Figures

1 Introduction

The Cloud-to-Thing Continuum is a novel cloud paradigm that is enabled by the seamless integration of cloud, edge, and Internet of Things (IoT) capabilities. This cloud model is defined as *"an extension of the traditional Cloud towards multiple entities (e.g., Edge, Fog, IoT) that provide analysis, processing, storage, and data generation capabilities"* [80]. The multitude of heterogeneous and interconnected devices continuously generates a large amount of data at the network's edge, significantly contributing to the advancement of next-generation services. Therefore, the Cloud-to-Thing Continuum holds the transformative potential to redefine various application domains, such as smart healthcare and smart cities [95].

However, this paradigm also brings unprecedented security challenges that need to be properly handled. The numerous interconnected devices engaged in data exchange and entities involved significantly broaden the security attack surface [121]. Traditional security measures tend to be static and rely on fixed configurations and rule-based approaches. There is no universal one-control-fits-all-scenarios. Static configurations, deployments, and management of security measures are insufficient to address risks within all possible situations. Certain configurations may work better under certain environmental conditions, while others may prove more effective when facing specific threat actors. Additionally, some controls are only relevant in specific scenarios, i.e., employees using specific devices or services. Therefore, traditional solutions are not designed to adapt to the dynamic and ever-evolving nature of the Cloud-to-Thing Continuum.

The lack of *adaptable security mechanisms* has led to the need to design and develop security solutions that can deal with evolving circumstances and scenarios [98]. Adaptable security refers to the ability of security measures, protocols, or systems to adjust, evolve, and respond to changing threats, vulnerabilities, and contextual information [17]. These mechanisms are designed to be flexible and responsive to the dynamic nature of cybersecurity. For example, an adaptable access control system should be able to dynamically

authorize or restrict access in response to evolving context information [92], i.e. a doctor who needs to access sensitive healthcare data to save a patient's life. It is worth noting that the concept of adaptability can be approached from various perspectives and levels.

Specifically, in this work, we focus on providing novel adaptable security mechanisms in the Cloud-to-Thing Continuum with particular emphasis on threat modeling and access control models, which we consider as two fundamental security techniques that can contribute to enhancing its security.

To build secure systems, security has to be taken into account from the early designing stages, adhering to security-by-design principles [43]. However, in most cases, this paradigm is not followed, and security operations are postponed until the end of the development phase. This problem is even worsened in the context under study, which is characterized by many heterogeneous entities. To address this challenge, we first propose a model-based threat modeling approach that enables the identification of threats and security controls in edge computing systems, a key component of the Cloud-to-Thing Continuum. Threats and security countermeasures are identified according to the properties of the assets and information of the data involved. Thus, the security profile dynamically adapts to the characteristics of the system components.

Then, we extended our approach to encompass cyber-physical systems (CPSs), which is another key technology of the Cloud-to-Thing Continuum ecosystem. This extension also considered dynamic moving target defense (MTD) techniques, which as for the original approach, are identified based on the asset properties and data transmitted/-stored. MTD methods increase the resilience of the systems by implementing strategies to proactively prevent or mitigate the realization of threats. Both of these approaches are implemented through a robust Threat Catalogue, which spans over 150 entries and is made available to the community.

Managing access to data, resources, and services is one of the major problems. Authorization permissions are usually implemented through access control policies. However, since the policies may change over time, we believe that analyzing dissemination models is crucial to evaluate how they can be effectively shared. Our empirical findings substantiate that distributed ledger technologies (DLTs) offer a robust solution for sharing policies in multi-region applications. Additionally, concerning the access control model, we introduce a novel risk-based authorization middleware capable of dynamically adapting authorization permissions in response to real-time contextual information. We applied

the proposed solution to healthcare scenarios, which is one of the sectors that has been significantly reshaped by the Cloud-to-Thing evolution. In this application domain, contextual information encompasses the patient's health status, inferred through federated learning (FL), a collaborative machine learning (ML) approach that allows leveraging a large amount of data while preserving their privacy. Each client trains a local model using its on-premises data and sends partial models to a parameter server, which aggregates to build a new global model. In the FL domain, we also present a framework that enables FL to increase the trustworthiness among unknown parties. Our approach provides adaptability since the contributions of participants are weighed according to their reputation score, and computed according to the quality of current and previous partial models. The comprehensive evaluation showcases that the proposed approach outperforms state-of-the-art solutions under different circumstances and datasets.

Finally, the Cloud-to-Thing Continuum comprises a large amount of IoT devices, which generate data crucial for informed decision-making and collaborating to deliver innovative services [100]. Therefore, to securely use such information, devices must be trustworthy. Most existing solutions are based on Public Key Infrastructure (PKI); however, digital identities are issued and managed by centralized authorities (CAs) that suffer from several concerns such as low scalability and limited interoperability. To overcome these challenges, the World Wide Web Consortium (W3C) has recently formalized and standardized decentralized identifiers (DIDs) and verifiable credentials (VCs). VCs present a decentralized approach to prove certain capabilities about entities, providing more fine-grain control. It is important to acknowledge that IoT devices may be susceptible to compromise or use components vulnerable to exploitation. Thus, such credentials have to be efficiently revoked to adapt to evolving conditions.

In this direction, we propose a novel revocation scheme based on an Elliptic Curve Cryptography (ECC)-accumulator. This approach facilitates efficient VC revocation while minimizing the burdens of storage and computational overhead. Moreover, it empowers devices to receive and apply updates even when disconnected from the network. We evaluated this revocation scheme across various deployment scenarios. Our experiments on commodity IoT devices demonstrate that each device only requires approximately 1.5 KB of storage to maintain verification information, half the storage required by the most efficient PKI certificates. Moreover, our experiments on hybrid networks, representing typical IoT protocols (e.g. Zigbee), show the latency in the order of milliseconds. Fi-

nally, large-scale analyses validated the scalability of our approach, even when up to 30% of the devices missed updates. Remarkably, nearly 90% of devices across the entire network received updates within the first hour, affirming the effectiveness of our approach in handling offline updates.

The remainder of this work is structured as follows. Chapter 2 provides an in-depth overview of cloud technologies, which lay the foundation of the Cloud-to-Thing Continuum, and provides the needed background on digital identities and DLTs, which are widely used in our work. In Chapter 3, we present a comprehensive discussion of adaptable security within the Cloud-to-Thing Continuum, exploring the critical concepts and principles. Then, Chapter 4 introduces security-by-design approaches, offering insights into the construction of a robust security framework for edge computing and CPSs. In Chapter 5, we present a risk-based authorization middleware for healthcare and a framework that allows trustworthy participation in FL. Chapter 6 presents an efficient revocation scheme specifically designed for VCs within IoT networks. Finally, Chapter 7 draws our conclusions.

Author's Publications

P1 - V. Casola, A. De Benedictis, C. Mazzocca, and R. Montanari, "Toward automated threat modeling of edge computing systems", IEEE International Conference on Cyber Security and Resilience (CSR), 2021.

P2 - E. Prezioso, F. Giampaolo, C. Mazzocca, A. Bujari, V. Mele, and F. Amato, "Machine Learning insights for behavioural data analysis supporting the Autonomous Vehicles scenario", IEEE Internet of Things Journal, 2021.

P3 - V. Casola, A. De Benedictis, C. Mazzocca, and R. Montanari, "Designing Secure and Resilient Cyber-Physical Systems: a Model-based Moving Target Defense Approach", IEEE Transactions on Emerging Topics in Computing, 2022

P4 - A. Sabbioni, C. Mazzocca, M. Colajanni, R. Montanari, and A. Corradi," A Fully Decentralized Architecture for Access Control Verification in Serverless Environments", IEEE Symposium on Computers and Communications (ISCC), 2022

P5 - C. Mazzocca, N. Romandini, M. Colajanni, and R. Montanari, “FRAMH: A Federated Learning Risk-Based Authorization Middleware for Healthcare”, IEEE Transactions on Computational Social Systems, 2022.

P6 - A. Sabbioni, C. Mazzocca, A. Bujari, R. Montanari, and A. Corradi, “A Decentralized Architecture for Dynamic and Federated Access Control Facilitating Smart Tourism Services”, Proceedings of the ACM Conference on Information Technology for Social Good, 2022.

P7 - C. Mazzocca, A. Sabbioni, R. Montanari, and M. Colajanni, “Evaluating Tangle Distributed Ledger for Access Control Policy Distribution in Multi-region Cloud Environments”, International Conference on the Quality of Information and Communications Technology, 2022.

P8 - A. Al Sadi, C. Mazzocca, A. Melis, R. Montanari, M. Prandini, and N. Romandini, “P-IOTA: A Cloud-Based Geographically Distributed Threat Alert System That Leverages P4 and IOTA”, Sensors, 2023

P9 - N. Romandini, C. Mazzocca, and R. Montanari, “Federated Learning Meets Blockchain: a Power Consumption Case Study”, 31st Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 2023.

P10 - C. Mazzocca, N. Romandini, M. Mendula, R. Montanari, and P. Bellavista, “TruFLaaS: Trustworthy Federated Learning as a Service”, IEEE Internet of Things Journal, 2023.

P11 - H. Batool, A. Anjum, A. Khan, S. Izzo, C. Mazzocca, and G. Jeon, “A Secure and Privacy Preserved Infrastructure for VANETs based on Federated Learning with Local Differential Privacy”, Information Sciences, 2024.

P12 - C. Mazzocca, N. Romandini, R. Montanari, and P. Bellavista, “Enabling Federated Learning at the Edge through the IOTA Tangle", Future Generation Computer Systems,

2024.

P13 - V. Casola, A. De Benedictis, C. Mazzocca, and V. Orbinato, “Secure Software Development and Testing: a Model-based Methodology”, Computer & Security, 2024.

Under Review

C. Mazzocca, A. Acar, S. Uluagac, and R. Montanari, “EVOKE: Efficient Revocation of Verifiable Credentials in IoT Networks”, under review in 33rd USENIX Security Symposium, 2024.

A. Feraudo, N. Romandini, C. Mazzocca, R. Montanari, and P. Bellavista, “DIVA: a DID-based Reputation System for Secure Transmissions in VANTEs Using IOTA", under review in Computer Networks, 2024.

C. Mazzocca, A. Acar, S. Uluagac, and R. Montanari, “A Survey on Decentralized Identifiers and Verifiable Credentials”, under review in IEEE Communication Surveys & Tutorials, 2024.

F. Magnanini, L. Ferretti, M. Andreolini, C. Mazzocca, and M. Colajanni, “SPOC: Survivable Passwordless Single Sign-On", under review in Computer & Security, 2024.

N. Romandini, A. Mora, C. Mazzocca, R. Montanari, and P. Bellavista, “Federated Unlearning: A Survey on Methods, Design Guidelines, and Evaluation Metrics”, under review IEEE Transactions on Neural Networks and Learning Systems, 2024.

2 Background

Cloud and distributed technologies have revolutionized conventional data storage, processing, and management methodologies. This chapter offers a comprehensive exploration of these technologies, laying the foundation of the Cloud-to-Thing Continuum. Furthermore, it also introduces digital identities and distributed ledger technologies, which we have widely used to provide adaptable access control.

2.1 Cloud Computing Models

In the ever-evolving realm of computing, traditional paradigms are being redefined by novel cloud modules, which comprise heterogeneous resources spread among different areas. This section presents four key modules that are at the forefront of this transformation and have been widely considered in our research.

2.1.1 Multi-Cloud

In the dynamic landscape of cloud computing, cloud providers empower customers with a versatile array of solutions to overcome availability and lock-in concerns. Multi-cloud models stand out as one of the most compelling approaches available to organizations. A multi-cloud strategy involves harnessing the power of multiple cloud services from various providers within a single, integrated infrastructure. One of the primary advantages of multi-cloud models is their ability to thwart vendor lock-in, which occurs when a company becomes overly reliant on a single cloud provider for all its cloud services. By adopting a multi-cloud approach, businesses liberate themselves from this vulnerability, ensuring uninterrupted operations even in the face of unexpected circumstances. Beyond resilience, the adoption of multiple cloud providers also brings cost optimization. Different cloud providers often present varying pricing structures for comparable services.

Leveraging this diversity, organizations can pick the most convenient options for their specific needs, thereby optimizing their expenditure. Moreover, multi-cloud models enable businesses to tap into the unique services and features offered by different providers, allowing them to tailor their cloud infrastructure to the precise requirements of each use case.

Furthermore, these models empower organizations to overcome geographical boundaries and deploy their applications strategically across multiple regions, offering valuable advantages that ripple through the entire cloud ecosystem. By distributing applications across multiple regions, multi-cloud models dramatically reduce network latency, creating an environment where data resides close to end-users. This enhances the user experience, where rapid data access and low response times become the norm.

As mentioned previously, one of the crucial advantages of multi-cloud models consists of its capacity to strengthen the resilience of applications. Indeed, the failure of a single region no longer leads to the catastrophic risk of rendering the entire service unavailable. Instead, applications continue to function seamlessly, drawing upon resources from operational regions, ensuring uninterrupted service delivery even in the face of localized disruptions. This resiliency is a key asset in today's digital landscape, where downtime can translate to substantial losses.

Moreover, the multi-region deployment strategy serves as a cornerstone for addressing the intricate challenges posed by data privacy laws and regulations. By offering the ability to store user-related data in specific geographic areas, multi-cloud models empower organizations to enhance data sovereignty and compliance. This strategic allocation of data not only ensures adherence to regional regulations but also fosters trust among users by emphasizing the sanctity of their personal information.

Despite the clear benefits, implementing a multi-cloud strategy can introduce complex security and management challenges that require to be properly handled [24]. A multi-region application must operate cohesively as a single service, adhering to consistent access control policies while maintaining a unified perspective on the involved data.

2.1.2 Edge and Fog Computing

Edge and fog computing offer distinctive approaches to handling computational tasks, particularly in environments that require proximity to data sources [54]. On the one

hand, edge computing emphasizes the decentralization of computing resources at the boundary of the network. Edge nodes play a key role in handling tasks such as data processing, storage, caching, and load balancing close to data sources. This approach minimizes latency and optimizes the user experience, making it particularly suitable for latency-sensitive applications like gaming platforms, autonomous vehicles, and real-time industrial automation. However, edge computing also brings inherent challenges. The diverse hardware platforms, operating systems, and communication protocols across edge devices necessitate robust management and coordination. Security concerns are heightened due to the distributed nature of edge nodes, which can have varying levels of security capabilities. Furthermore, the management of personal and sensitive data in edge environments requires meticulous attention to privacy and compliance.

On the other hand, fog computing is an extension of the edge paradigm that introduces an additional layer of computational resources between the edge and the cloud. Fog nodes, situated closer to the edge, are responsible for processing and filtering data before relaying it to the cloud. This intermediary layer offers benefits such as improved data analysis and reduced data transfer to the cloud, reducing bandwidth costs and latency. It is particularly well-suited for applications that require real-time data analysis and decision-making. For instance, in a smart city context, fog nodes can process sensor data locally to optimize traffic management or enhance public safety. Nonetheless, like edge computing, fog computing also presents its own set of challenges. The integration and orchestration of fog nodes within a complex network architecture can be intricate. Ensuring the security of data and applications across fog nodes is essential, as these nodes often serve as gateways to the cloud. Additionally, maintaining a consistent and up-to-date software and policy framework across a distributed fog network is a non-trivial task.

While edge and fog computing have their respective merits and challenges, they are not mutually exclusive. Indeed, they can complement each other effectively. Edge nodes serve as the first line of defense for quick data processing, while fog nodes further refine and filter data before relaying it to the cloud for deep analysis and long-term storage. This synergistic approach combines the advantages of both paradigms, enhancing overall system performance and efficiency.

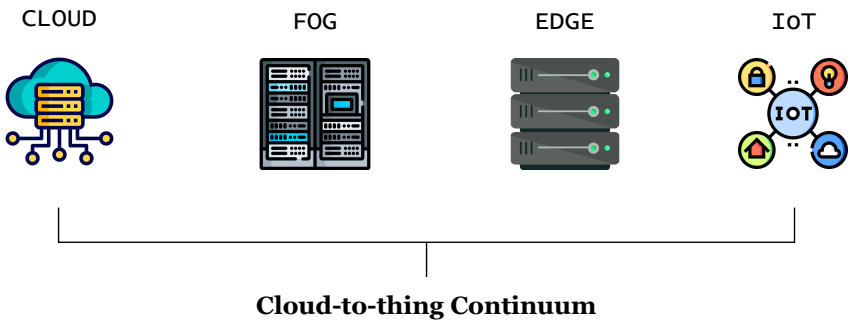


Figure 2.1: Cloud-to-Thing Continuum.

2.1.3 Cloud-to-Thing Continuum

In the ever-expanding landscape of the IoT and cloud computing, the concept of the Cloud-to-Thing Continuum emerges as a transformative and dynamic framework. As shown in Figure 2.1, the continuum is given by the seamless integration of cloud-based services and IoT devices, enabling a bidirectional flow of data, control, and intelligence that transcends the traditional boundaries of both cloud and IoT ecosystems. The Cloud-to-Thing Continuum leverages the convergence of two powerful domains: the cloud infrastructure and the distributed, sensor-laden, and real-world impact of IoT devices. This integration paves the way for a new era of connectivity, where data generated by IoT devices is not simply sent to the cloud for storage and analysis but is also utilized to inform and control the devices themselves, fostering a two-way communication paradigm. Therefore, the paradigm described above and other technological advancements like 5G and ML contributed to its widespread.

This transformation promises to reshape industries, from smart cities and healthcare to manufacturing and agriculture, by enabling data-driven decision-making, automation, and responsiveness at a scale and precision previously unattainable. However, this newfound potential also brings forth a multitude of challenges, including data security, privacy, real-time processing, and scalability, all of which must be carefully navigated to harness the full power of such a paradigm.

2.2 Digital Identity

The recent expansion of digital services accessible through network connections has led to an unprecedented increase in the creation of digital identities. Virtually every person on the planet possesses at least one digital identity, granting them access to these online resources. It is important to recognize that digital identities extend beyond human individuals; they encompass a wide array of entities, ranging from objects to entire organizations. This is particularly relevant in the Cloud-to-Thing Continuum landscape, characterized by a vast amount of entities that require proper identification. In the following, we delve into the evolution of digital identities, paying special attention to emerging technologies such as decentralized identifiers and verifiable credentials, which are the novel decentralized standards to identify and authorize entities.

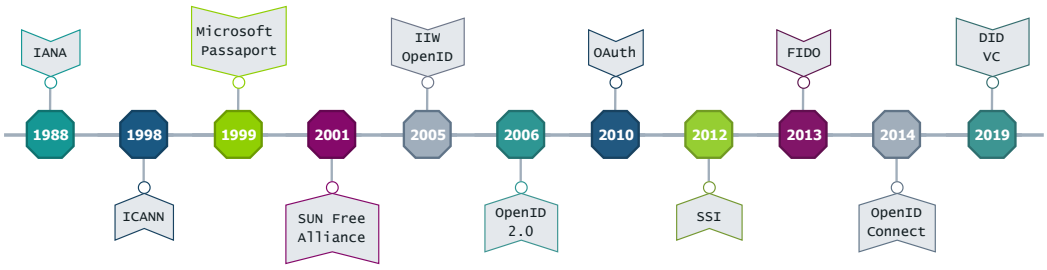


Figure 2.2: Timeline of digital identity evolution.

2.2.1 Digital Identity Evolution

The digital identification of entities within networks and systems has always been one of the major concerns since the early days of the Internet. Over the years, we have been witnessing different digital identity eras, each of them characterized by various models and technologies. This section goes through the digital identity evolution, discussing how identification has gradually shifted from centralized to decentralized models [6]. Figure 2.2 sketches the digital identity evolution over time.

Centralized Identity

In centralized identity management approaches, central authority plays a key role in handling digital identities and their information. One of the first forms of digital identifica-

tion dates back to 1988 when the Internet Assigned Numbers Authority (IANA) took on the responsibility of validating IP addresses. Subsequently, we saw the emergence of organizations like the Internet Corporation for Assigned Names and Numbers (ICANN), which oversaw the legitimacy of domain names. At the heart of this era of centralized identities, users were identified through the conventional combination of usernames and passwords.

However, centralized approaches are not without their flaws. They suffer from a glaring vulnerability - a single point of failure. In these systems, all user-related information, including sensitive data, is entrusted to centralized entities. This raises significant concerns, particularly in the context of evolving privacy and legal regulations such as the European General Data Regulation Protection (GDPR) [41].

Furthermore, the use of password-based methods adds another layer of complexity. Users, often prioritizing simplicity over security, tend to choose weak passwords, leaving them susceptible to various attacks such as a dictionary or spray attacks. For example, an attacker may be able to acquire the user's password from an unsecured service (e.g. ordering food from her favorite restaurant) and reuse it to gain access to more valuable services, such as a bank account.

Federated Identity

Federated identity represents a groundbreaking concept that empowers users to wield the same identity credentials across a multitude of websites and applications. This streamlined approach alleviates users from creating distinct accounts for each system, which can be a challenge for usability since users are demanding to remember a high number of usernames and passwords. Federated approaches rely upon the establishment of trust among various entities that facilitate secure user authentication and authorization across disparate systems.

The inception of federated identity can be attributed to big companies such as Microsoft, which introduced its Passport program. This initiative enabled users to navigate various websites seamlessly through a single, unified login. In a parallel development, Sun Microsystems catalyzed the formation of the Liberty Alliance, also known as the Liberty Federation, in 2001. This consortium was dedicated to crafting open standards that underpinned federated identity and identity-driven web services. In more recent times, IT

giants like Google and Facebook have embraced the concept of single sign-on (SSO) support, further validating the efficacy of federated models.

However, it is worth noting that while federated identity models have made significant progress in mitigating concerns related to identity fragmentation, a certain degree of centralization persists within individual websites. Such a centralization issue limits the extent of mutual recognition among large-scale systems, reminding us that the journey toward comprehensive federated identity solutions is still evolving.

User-centric Identity

The growing demand for greater control over personal data has led to the era of user-centric identity, where individuals exert the power to independently manage their identities, whether it be on a singular platform or across multiple authorities, all without the need for a centralized federation. In this paradigm, users exercise absolute autonomy, retaining full control and necessitating explicit consent for any sharing or modifications to their personal data. This shift ensures a major emphasis on privacy and security.

A pivotal moment in this trajectory occurred in 2005, when the Identity Commons, a significant American organization dedicated to supporting, facilitating, and promoting an open identity layer for the Internet, played a transformative role in the inception of the Internet Identity Workshop (IIW). The IIW positions users as the central figures in the realm of identity management, with the aim to empower individuals in shaping their online identities. Notably, it has laid the foundation for numerous illustrious projects that embody the principles of user-centric identity, including OpenID, OpenID 2.0, OpenID Connect, OAuth, and FIDO. Underpinning these user-centric approaches is the concept of individuals wielding authority over their data, enabled by a multitude of authenticators such as JWT and certificates issued by diverse service providers. These digital assets are securely stored within users' personal devices, granting them unparalleled control over their online identities.

Nevertheless, it is crucial to acknowledge that user-centric identity systems, while granting substantial empowerment to individuals, still hinge on trust relationships with service providers and authorities. Users must place their trust in these entities, banking on their responsible and ethical handling of personal information, free from misuse or exploita-

tion. This element of trust underscores the ongoing evolution of user-centric identity systems, with continuous efforts aimed at enhancing user data protection and privacy.

Self-Sovreign Identity

Self-Sovereign Identity (SSI) represents the last era of digital identity development. This groundbreaking concept made its debut in 2012, representing a remarkable advancement over user-centric identity systems. At its core, SSI offers users the ultimate authority, allowing them complete control over their identities. Indeed, users decide when, if, and how they wish to disclose or alter their personal data. The robustness of a secure SSI system hinges on its reliance on cutting-edge technologies, with a decentralized infrastructure forming its sturdy foundation. DLTs are one of the enabling technologies in shaping the architecture of SSI systems, which effectively remove any need for central authorities, creating an environment of trustlessness. Within this ecosystem, users interact with services and applications without ceding their sensitive information to a single entity. The decentralized framework ensures that data remains cryptographically safeguarded and verifiable, establishing transparency and immutability in identity management.

In the SSI landscape, each individual possesses a DID, which they directly control. To validate their attributes or claims, users issue VCs. These credentials can be cryptographically verified by anyone, eliminating the necessity to rely on a centralized authority for verification. Embracing SSI marks a crucial paradigm shift in the conceptualization and implementation of digital identity. By placing users squarely in the driver's seat of their identity management journey, SSI ushers in a profound sense of self-sovereignty. Individuals can now navigate the digital terrain with unwavering confidence and unprecedented autonomy, fundamentally transforming the landscape of digital identity into a realm where individuals have full power over their data.

2.2.2 Decentralized Identifiers

DIDs have emerged as a groundbreaking form of identifier, gaining widespread acceptance within decentralized identity systems. The W3C played a pivotal role in formalizing and standardizing DIDs, dedicating substantial resources and collaborative efforts from 2017 to 2019. This concerted effort culminated in the official recognition of the DID

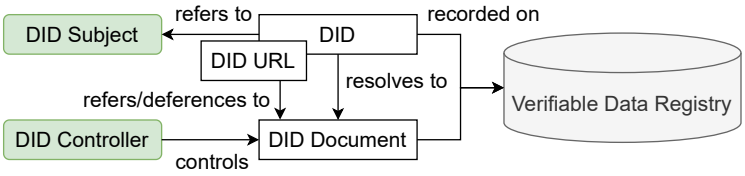


Figure 2.3: Overview of a DID-based architecture and the relationship of the main components.

specification as a W3C Recommendation. Fig. 2.3 provides a concise overview of the key components of a DID-based architecture.

A DID serves as a unique identifier for a *DID Subject*, which can be a human or a non-human entity. Comprising three essential components, a DID is a textual string that includes the Uniform Resource Identifier, the identifier specific to the *DID method*, and the method-specific identifier for the DID. A *DID URL* enhances a basic DID by incorporating additional standard URI components like path, query, and fragment. This extension allows for precise resource location, such as a cryptographic public key within a *DID Document* or an external resource linked to the DID Document. Each DID seamlessly resolves to a DID Document, a machine-readable JSON-LD document containing information about the DID Subject. A DID Document encompasses cryptographic public keys, service endpoints, authentication parameters, timestamps, and supplementary metadata. DIDs have been meticulously designed to eliminate the need for identity providers and centralized authorities. An entity can substantiate its ownership of a DID by utilizing the private key corresponding to the public key embedded in the DID document. To verify this proof, the verifier merely needs access to the publicly shared DID Document, typically stored in a *Verifiable Data Registry*. Fig. 2.4 provides a tangible example of a DID and its corresponding DID Document. The *DID Controller* denotes the entity vested with the authority to modify the DID Document. It is noteworthy that a single DID may have multiple controllers, and as illustrated in the example, it can coincide with the DID Subject.

The DID specification introduces three distinct types of DIDs:

- *Anywise*: A DID can be employed with an unspecified number of parties, often strangers. This versatility allows for widespread use without constraints on the number of relationships that can be established.

did:example:123456789abcdefghi

```
{
  "@context": [
   "https://www.w3.org/ns/did/v1",
   "https://w3id.org/security/suites/ed25519-2020/v1"
]
"id": "did:example:123456789abcdefghi",
"authentication": [{
  "id": "did:example:123456789abcdefghi#keys-1",
  "type": "Ed25519VerificationKey2020",
  "controller": "did:example:123456789abcdefghi",
  "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZp..."
  }]
}
```

Figure 2.4: An example of decentralized identifier and DID Document.

- *Pairwise*: A DID is known exclusively by its subject and precisely one other party, typically a service provider. Pairwise DIDs address privacy concerns by ensuring that each relationship has a unique DID, mitigating the risk of correlation between different interactions.

- *N-wise*: A DID is intended to be known by strictly N parties, including the subject. This is a more general concept that encompasses pairwise DIDs as a special case when N equals 2.

DIDs are resolved through a universal resolver capable of supporting multiple DID systems. For a DID system to align with the universal resolver, it must implement a DID adaptor, serving as an interface between system-specific DID methods and the universal resolver.

Furthermore, we have also witnessed the birth and widespread on novel DID-based protocols. In this direction, The DID Auth protocol empowers an identity owner to utilize their client application, such as a mobile device or browser, to prove their control over a DID to a service provider. This protocol operates on a challenge-response cycle that can be customized according to the specific context. By leveraging the DID Auth protocol, conventional authentication methods like usernames and passwords can be replaced, forging an authenticated communication channel between the identity owner and the service provider. The DID Comm protocol is another DID-based protocol that facilitates private and secure communication among two or more SSI entities in a peer-to-peer

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/1872",
  "type": ["VerifiableCredential", "AlumniCredential"],
  "issuer": "https://example.edu/issuers/565049",
  "issuanceDate": "2010-01-01T19:23:24Z",
    "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "alumniOf": {
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      "name": [{
        "value": "Example University",
        "lang": "en"
      }, {
        "value": "Exemple d'Université",
        "lang": "fr"
      }]
    }
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2017-06-18T21:19:10Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod":
"https://example.edu/issuers/565049#key-1",
    "jws":
"eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0..."
  }
}
```

Figure 2.5: Example of a verifiable credential.

fashion. This protocol relies on DIDs and enables mutual authentication between the participating parties, ensuring the confidentiality and security of their interactions.

2.2.3 Verifiable Credentials

VC is another specification developed by the W3C with the primary aim of creating an interoperable data structure that can represent claims in a way that is both cryptographically verifiable and tamper-proof. In essence, VCs serve as a powerful tool for establishing trust in online interactions. Fig. 2.5 provides a practical example of a VC, illustrating how it can enable all alumni of "Example University" to receive discounts on season tickets for sporting events.

Within the VC ecosystem, key roles come into play. A *holder* is an entity with control over one or more VCs, essentially the custodian of these credentials. Conversely, an *is-*

suer is an entity responsible for creating and issuing new verifiable credentials. Examples of credential issuers can range from financial institutions like banks to government agencies. A *verifier*, on the other hand, is an entity that seeks to verify the authenticity and validity of one or more VCs. Consider an e-commerce website that requires customers to provide credentials; it acts as a verifier in this context. The dynamics of these roles form the foundation of trust in the VC ecosystem, enabling secure and transparent interactions. Fig. 2.6 offers an overview of these critical actors within the realm of verifiable credentials.

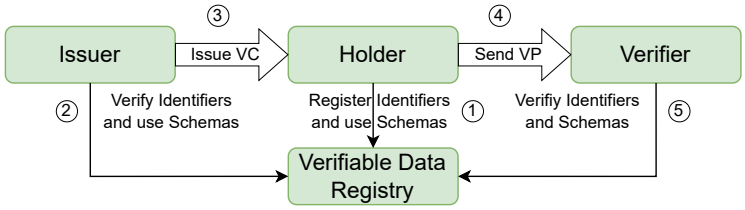


Figure 2.6: Overview of VC main actors.

To facilitate the creation and verification of identifiers, cryptographic keys, verifiable credential schemas, and other pertinent data essential for utilizing VCs, a verifiable data registry plays a pivotal role. This registry serves as a mediator, ensuring the smooth operation of the ecosystem and acting as a reference point for various entities.

A typical VC comprises several essential elements, including the Uniform Resource Identifier (URI) of the subject, the URI of the issuer responsible for the claims, and URIs that uniquely identify the credential. Additionally, a verifiable credential includes conditions for claim expiration and cryptographic signatures for enhanced security. The URI can also take the form of a DID, providing flexibility and self-sovereignty to the subject of the credential. It's worth noting that the W3C Verifiable Credentials Working Group has introduced the concept of verifiable presentations (VPs), which specify methods for signing and presenting VCs by the holder. VCs or VPs can be described using various formats, including JSON-LD, JSON, or JSON Web Tokens (JWT). Fig. 2.7 illustrates a VP obtained from the VC presented in the previous example.

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "type": "VerifiablePresentation",
  "verifiableCredential": [{
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://www.w3.org/2018/credentials/examples/v1"
    ],
    "id": "http://example.edu/credentials/1872",
    "type": ["VerifiableCredential", "AlumniCredential"],
    "issuer": "https://example.edu/issuers/565049",
    "issuanceDate": "2010-01-01T19:23:24Z",
    "credentialSubject": {
      "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
      "alumniOf": {
        "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
        "name": [{"value": "Example University", "lang": "en"},
          {"value": "Exemple d'Université", "lang": "fr"}]
      }
    },
    "proof": {
      "type": "RsaSignature2018",
      "created": "2017-06-18T21:19:10Z",
      "proofPurpose": "assertionMethod",
      "verificationMethod": "https://example.edu/issuers/565049#key-1",
      "jws": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0..."
    }
  }],
  "proof": {
    "type": "RsaSignature2018",
    "created": "2018-09-14T21:19:10Z",
    "proofPurpose": "authentication",
    "verificationMethod": "did:example:ebfeb1f712ebc6f1c276e12ec21#keys-1",
    "challenge": "1f44d55f-f161-4938-a659-f8026467f126",
    "domain": "4jt78h47fh47",
    "jws": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0..."
  }
}
```


Figure 2.7: Example of verifiable presentation.

2.2.4 Verifiable Data Registry

In the context of DIDs and VCs, the role of a verifiable data registry emerges as a pivotal and indispensable element. This registry assumes a crucial role in facilitating the creation, management, and verification of various components, including identifiers, cryptographic keys, credential schemas, and other essential data essential for the effective utilization of these decentralized identity technologies.

A verifiable data registry functions as a trusted intermediary within the ecosystem of DIDs and VCs- It assumes the role of a custodian, a repository, and a reliable source of vital information. It serves as a repository or database that stores and provides access to essential information related to DIDs and VCs. This includes the storage of DID Documents, which contain information about DIDs, such as cryptographic public keys, ser-

vice endpoints, authentication parameters, timestamps, and metadata. One of the registry's paramount functions lies in its ability to enable the verification of DIDs and VCs by providing a continuously available source of trust. When a verifier has to validate a DID or VC, it interacts with the registry to collect the necessary information. By accessing the public information stored in the registry, the verifier can validate the authenticity, integrity, and validity of the DIDs and VCs involved in the verification process. This allows for the establishment of trust and confidence in the decentralized identity ecosystem.

Moreover, the verifiable data registry assumes a fundamental role in the orchestration of the lifecycle of DIDs and VCs. Entities looking to create new DIDs or issue fresh VCs engage with the registry, ensuring the proper registration and management of associated information. Similarly, when a DID or VC needs to be revoked or updated, the verifiable data registry can handle the necessary operations to reflect the changes in the system. Such a registry also offers interoperability and standardization. It guarantees consistency, enforcing uniform data formats, validation rules, and data-sharing protocols. Thus, it ensures that the information stored within it respects standards and specifications. This commitment to conformity fosters compatibility and seamless integration, bridging gaps and forging connections across different DID methods, VC issuers, and verifiers.

It is important to note that while the W3C's standards do not prescribe a specific implementation for this registry, many proposed approaches are rooted in DLTs. Nevertheless, alternative solutions have also emerged, such as Information-Centric Networking (ICN), a novel networking paradigm that offers a compelling avenue for realizing the registry's potential [5].

2.3 Distributed Ledger Technologies

DLTs have emerged as a groundbreaking innovation with the potential to revolutionize numerous industries, primarily due to their remarkable security features. These technologies offer a robust and transparent way to record and verify transactions and data across a network of decentralized nodes. Instead of relying on a single, central authority for data control, DLTs distribute the information across a network of nodes. This decentralization ensures that every node maintains a coherent copy of the data, fostering a collaborative ecosystem where multiple entities can participate without the need for in-

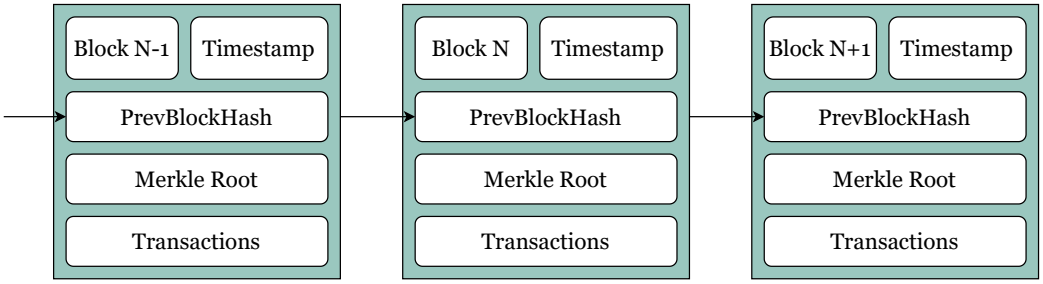


Figure 2.8: The blockchain data structure.

termediaries or third parties. One of the key features of DLTs relies upon their capacity to be resilient against tampering. Contrary to traditional data sources, i.e. databases, DLTs use an append-only data structure. Therefore, the data recorded is immutable, making it impervious to any form of tampering or manipulation. These systems rely on decentralized Peer-to-Peer (P2P) networks, avoiding the vulnerabilities associated with a centralized control entity and the risks of a single point of failure. To maintain the integrity and synchronization of the network, DLTs rely on consensus protocols grounded in strong cryptographic principles. These protocols establish a trust framework among participants, enabling them to collectively validate and agree upon the validity of transactions and data updates. This cryptographic foundation instills confidence in the reliability of the system, as participants can be assured that only authorized and verified changes are accepted into the ledger.

2.3.1 Blockchain

Blockchain is certainly the most popular data structure among DLTs, as proven by the numerous research studies and commercial applications that adopt it. The concept was introduced for the first time in 2008 by Satoshi Nakamoto [81]. A blockchain is a chain of blocks and each block comprises tamper-proof information, whose integrity is guaranteed through cryptographic techniques (i.e., hash and digital signature). Any modification to data produces a completely different hash, making it impossible to tamper stored information. Figure 2.8 sketches the blockchain data structure.

Consensus protocols are the heart of blockchain technology, ensuring that distributed networks of nodes can agree on the state of the blockchain, validate transactions, and maintain the integrity of the ledger. Achieving consensus in a decentralized and trust-

less environment is one of the fundamental challenges that blockchain protocols seek to address. Several consensus mechanisms have been developed, each with its own set of characteristics and trade-offs. However, the most popular solutions are currently represented by *proof of work* (PoW) and *proof of stake* (PoS). PoW is the consensus mechanism that underlies Bitcoin, which is considered the original cryptocurrency. In PoW, miners compete to solve complex mathematical puzzles, with the first one to solve it having the right to add a new block of transactions to the blockchain. This process, known as *mining*, requires significant computational power and energy, making it secure but energy-intensive. PoW has proven its robustness in terms of security and decentralization but has raised environmental concerns due to its energy consumption. that has a significant impact in terms of energy and power consumption [11]. PoS is an alternative energy-efficient consensus mechanism. In PoS, validators are chosen to create new blocks and validate transactions based on the amount of cryptocurrency they hold and are willing to *stake* as collateral. Validators are incentivized to act honestly because their staked assets may be lost if they misbehave. The adoption of PoS has experienced a remarkable surge in popularity in recent years, with Ethereum's official switch to a PoS consensus mechanism in 2022 marking a significant milestone in this transformative trend [40]. This shift reflects the growing recognition of PoS as a more energy-efficient and sustainable alternative to traditional PoW systems, as blockchain ecosystems seek to balance scalability, security, and environmental considerations in their quest for innovation.

Recently, an increasing number of blockchains have started integrating *smart contracts*, which are self-executed applications coded in blockchain networks. Thus, trust and transparency are not only assured but automated. executed on the blockchain. Smart contracts eliminate the need for intermediaries, allowing individuals and organizations to engage in agreements with the confidence that contractual obligations will be fulfilled as programmed. They are extremely versatile, indeed, they are not confined to financial applications alone. While they indeed revolutionize lending, trading, and financial services through decentralized finance (DeFi), they also find homes in supply chains, healthcare, voting systems, and beyond. It is worth noting that smart contracts do not operate in isolation. They often require access to external data, such as market prices, weather conditions, or real-world events, to function effectively. This is where decentralized oracle networks come into play. These networks serve as bridges between the blockchain and the external world, providing smart contracts with reliable, tamper-resistant data inputs.

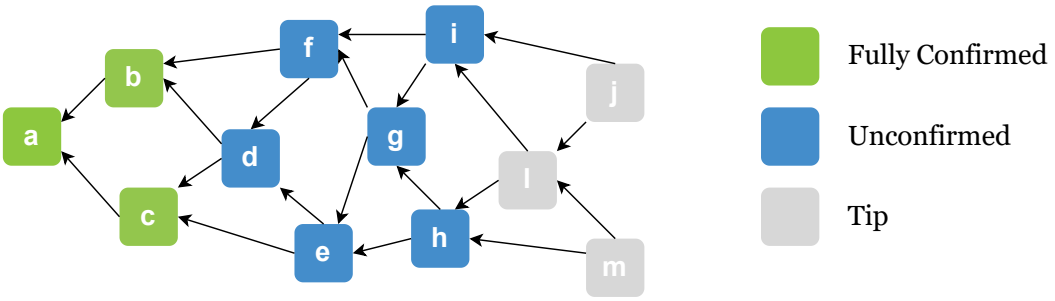


Figure 2.9: The Tangle data structure.

Through decentralized oracle networks, smart contracts can make decisions based on real-time information without compromising the trustless nature of blockchain technology. However, smart contracts, like any technology, are not immune to bugs or vulnerabilities. Legal and regulatory frameworks are still grappling with the implications of self-executing contracts, raising questions about liability and dispute resolution in this brave new world.

2.3.2 Directed Acyclic Graph (DAG)-based Ledger

Directed Acyclic Graph (DAG)-based ledger, DAG for briefness, represents a novel approach that has garnered significant attention in recent years. Contrary to traditional blockchain systems that rely on linear, sequential blocks, DAGs employ a fundamentally different structure, aiming to address some of the scalability and efficiency limitations associated with blockchains. In a DAG, transactions are linked to multiple predecessor, creating a branching and interconnected network. This structure allows transactions to be processed in parallel, significantly enhancing the potential for scalability and throughput. DAGs leverage concurrency, enabling multiple transactions to be confirmed simultaneously. This departure from the linear block structure of traditional blockchains can potentially result in faster transaction processing and higher overall throughput. Transactions in a DAG can be validated asynchronously by different nodes. This means that consensus can be reached without requiring all nodes to validate every transaction, reducing latency. Since DAGs do not rely on energy-intensive mining processes like PoW they offer the promise of reduced energy consumption, aligning with growing concerns about the environmental impact of blockchain technologies.

IOTA [91] is the most popular DAG and it is designed specifically for the IoT. IOTA adopts a unique data structure called Tangle, whose data structure is shown in Figure

2.9. Transactions are validated by the nodes they are connected to, allowing for fast performance without the need for middlemen such as miners or validators. The Tangle also allows for zero-value transactions, which do not require validation by network participants since they do not involve any transfer of value. As a result, they can be attached to the Tangle without the risk of double spending, significantly reducing the time required to share information. As blockchain, IOTA networks can be deployed as private or public. In addition, IOTA distinguishes clients and nodes. On the one hand, a client is any entity (i.e., human or not) that submits transactions to a node. On the other hand, a node is responsible for verifying their correctness and attaching them to the Tangle. An IOTA network also comprises additional node types named *Coordinator* and *Permanode*. There is only a Coordinator that regularly produces *milestones*, trusted signed transactions used by nodes to confirm transactions. The signature guarantees that nobody can fake the signatures on milestones, thus, they are always legit. In particular, a transaction is confirmed only when directly or indirectly referenced by a milestone that nodes have validated. It is worth noting that the use of the Coordinator is temporary since it will be removed in incoming updates. On the other hand, Permanodes are responsible for keeping the history of all the transactions that occurred. Such a component is particularly relevant in specific scenarios since nodes may be constrained devices that cannot memorize the entire Tangle, thus periodically deleting recorded transactions through a pruning operation.

3 Adaptable Security in the Cloud-to-Thing Continuum

Cloud-to-Thing Continuum environments comprise many heterogeneous technologies and resources that operate in a continuously evolving landscape. This high dynamicity demands novel security solutions that should be dynamically adaptable to different circumstances and scenarios. However, such a concern can be addressed from different perspectives and levels. This dissertation proposes novel solutions for threat modeling and access control, which are valuable security approaches to making the Cloud-to-Thing Continuum more secure. Specifically, the thesis presents a threat modeling methodology that adapts to the system features and provides multiple contributions to adaptable access control. It includes the evaluation of policy distribution, novel frameworks that dynamically modify their behavior, and authorization revocation.

This chapter discusses the main limitations of traditional solutions and introduces the concepts of threat modeling and access control. Furthermore, it also reviews existing approaches closer to the security mechanisms presented in this thesis.

3.1 Adaptable Security Mechanism

The integration of cloud, edge, and IoT has remarkably contributed to the birth and widespread of the Cloud-to-Thing Continuum. This paradigm has the potential to significantly contribute to the designing and developing of next-generation services. For example, the integration of the Internet of Medical Things (IoMT), edge, and cloud services can be used to remotely monitor the health condition of a patient and collect vital physiological data [50], which are helpful for emergency medical decisions and chronic disease detection. In such a scenario, caregivers assistance anywhere and anytime. These environ-

ments are extremely dynamic and comprise many heterogeneous resources, which makes traditional security mechanisms unsuitable for several reasons.

Specifically, existing approaches are not designed to handle a large number of interconnected devices and services. They rely on static rules and configurations that struggle to keep up with the dynamic landscape of the Cloud-to-Thing Continuum. Such solutions are not able to adapt their behavior to different circumstances and scenarios. It is worth noting that there are no one-control-fits-all-scenarios, and no static configuration, deployment, use, or management of security control can cope with risk within all possible situations [17]. Some configurations can work better under certain environmental conditions, others will be more secure when facing specific threat actors, and some of them will be only used when employees use specific devices or services.

However, to deal with the dynamicity and heterogeneity of the Cloud-to-Thing Continuum, we demand novel and adaptable security solutions capable of adjusting their behavior and security controls by the current context [104]. An adaptive system can modify its composition, architecture, or behavior in response to its state and its perception of the external environment.

It is worth noting that the concept of adaptable security encompasses a broad spectrum of security solutions that can be approached from various perspectives. For example, an adaptable system may employ authentication mechanisms based on the resources of the involved entities. Another example of adaptability can be observed in how access control policies are shared across different areas; in some scenarios, prioritizing high availability might be the optimal choice, while in others, strict consistency could be the preferred approach. Thus, a truly effective adaptable system should be adept at selecting the best security mechanism according to the current context. In this chapter, we introduce threat modeling and access control which are the security techniques that have been widely investigated in this thesis. Moreover, we also revise the current state-of-the-art works that are closer to the solutions presented in the following chapters.

3.2 Threat Modeling

Threat modeling is a fundamental practice that should be always followed to design secure systems. This concern is even exacerbated in the Cloud-to-Thing Continuum due

to the huge amount of entities involved. Indeed, it is undeniable that having several resources that interact and perform operations brings new and unprecedented cyber risks [4]. Typical security concerns related to the loss of control experienced in cloud environments add up with the issues affecting the edge and device layers, characterized by highly heterogeneous hardware platforms, operating systems, and communication protocols, different computation, storage, and communication capabilities, typically coarse-grained access control mechanisms, and a general lack of attack awareness due to the limited interfaces usually offered by devices. To make things worse, such technologies are widely employed in applications that collect and process personal and sensitive data, whose protection should be ensured at any level, independently of the specific capabilities of involved devices. In this context, identifying the actual threats that affect a given deployment is a challenging task, as risks vary significantly by individual use case and application domain, and heavily depend on involved components and technologies. Unfortunately, despite increasing interest by both the academic and industrial communities, there is still no discernible consensus on security best practices, and activities like threat analysis and countermeasure selection are entirely left to security experts.

However, traditional threat model approaches are static. They usually identify threats associated with a given asset and corresponding security countermeasures to mitigate them, without adapting to the features of the assets that compose the systems. Furthermore, they do not provide any reference to alternative techniques to implement in case the attacker breaches the system defeating enforced security controls. Therefore, they do not consider how to react to undesired situations. In this direction, moving target defense represents a valuable solution to increasing the resilience of the Cloud-to-Thing Continuum. MTD is a proactive cyber-defense paradigm according to which a system's attack surface is continually changed over time using reconfiguration, to increase complexity and cost for attackers during reconnaissance activities, limit the exposure of vulnerabilities, and increase overall system resilience.

3.2.1 Related Work

There are just a few approaches in the literature that focus on the secure design and analysis of the IoT and edge layers. IoT Sentinel [78] is a system capable of automatically identifying the types of devices and subsequently establishing which rules should be enforced

to constrain the communications of devices affected by potential security vulnerabilities. This security system allows for minimizing the damage resulting from vulnerable devices. A framework for modeling and assessing the security of IoT was proposed in [47]. It is employed to build a graphical security model aiming at capturing potential attack paths in the network. Furthermore, the authors provide a security evaluator responsible for automating the security analysis. The results of the assessment of the security level of the IoT network provide a clearer picture of which assets and paths should be protected at first. Then, the defense strategies are compared to choose the most effective device-level security strategies. Soteria [26] is a static analysis system proposed to validate whether an IoT application or IoT environment adheres to identified security, safety, and functional properties. It translates platform-specific IoT source code into an intermediate representation and then extracts a state model that verifies the desired properties. In [15], the authors propose an IoT Security Model (IoTSM) that allows organizations to plan and implement a strategy for developing end-to-end IoT security. This approach also enables analyzing, describing, and measuring the security posture, level, and practice of an IoT organization. Most of the current state-of-the-art research efforts target security issues, challenges, and frameworks for securing edge computing systems [4], whereas the area of automated threat modeling is still in its infancy.

To overcome these challenges, Chapter 4 proposes a semi-automated threat modeling fine-grained methodology for edge computing systems that identifies threats and security controls according to the features of the system. Furthermore, this chapter also extends the methodology to CPS, the extended approach also encompasses MTD techniques. To the best of our knowledge, we are the first to propose a semi-automated threat modeling approach for CSPs that explicitly takes into account the data flow and some relevant features of the components to determine threats as well as their security mitigation and alternative security techniques.

3.3 Access Control

Access control is an essential security mechanism that determines who is permitted to access various services, resources, and data within a system. Access control encompasses

four primary models, each distinguished by its approach to managing access to sensitive resources and information:

- *Discretionary access control (DAC)*: In DAC models, each object within a protected system boasts an owner who possesses the authority to grant access to users at their discretion. This model offers a case-by-case control over resources.
- *Mandatory access control (MAC)*: MAC models are based on the allocation of access through clearance levels. A centralized authority oversees access rights and organizes them into hierarchical tiers, systematically expanding in scope. This model finds widespread use in government and military domains due to its stringent control mechanisms.
- *Role-based access control (RBAC)*: Within RBAC models, access rights are granted based on predefined business functions, rather than the individual identities or seniority of users. The primary objective is to provide users with precisely the data required to fulfill their job responsibilities, mitigating the risk of overexposure.
- *Attribute-based access control (ABAC)*: ABAC models introduce flexibility into access grants by considering a combination of attributes and environmental conditions, such as time and location. ABAC stands as the most finely-grained access control model, streamlining the management of role assignments and enhancing precision in resource access.

3.3.1 Adaptable Access Control

Due to its key features, ABAC is considered highly suitable for adaptable access control in the Cloud-to-Thing Continuum [7]. ABAC provides a fine-grained approach to access control, allowing administrators to define access policies based on a wide range of attributes and contextual factors. This granularity enables precise control over who can access specific resources and under what conditions, making it adaptable to various scenarios. This is particularly relevant in Cloud-to-Thing Continuum environments that are inherently dynamic, with resources being provisioned, scaled, and de-provisioned as needed. ABAC's ability to consider dynamic attributes such as time, location, user attributes, and resource properties makes it well-suited for adapting access control policies

to changing circumstances. In these contexts, scalability is crucial. ABAC models scale effectively because it does not rely on predefined roles or clearances. Instead, access policies can be adjusted to accommodate a growing number of users, resources, and changing business requirements. Furthermore, ABAC allows administrators to create flexible access control policies without the need for complex role management. This flexibility is vital in cloud environments where access requirements can vary widely, and traditional role-based models may become cumbersome to manage. Cloud-based environments face evolving threats and vulnerabilities. ABAC can adapt to emerging security challenges by considering attributes related to security, such as device trust level or user behavior. This adaptability enhances the security posture of the cloud continuum. Finally, ABAC can streamline access control administration by reducing the need for constant role updates and manual access adjustments. It automates access decisions based on attributes, saving time and reducing the risk of human error.

Risk-based Access Control

Risk-based access control stands as a highly valuable approach to ensure robust adaptability, especially in the face of unpredictable circumstances and varying conditions. This dynamic access control methodology transcends the constraints of conventional authorization methods by adapting access decisions based on the assessed level of risk accompanying each access request. Risk-based access control operates as a proactive safeguarding mechanism, allowing organizations and systems to respond dynamically to ever-evolving threats and challenges. Continuously evaluating the potential risks tied to each access attempt, empowers decision-makers to make informed choices that balance security and operational flexibility. This approach to access control is a testament to the agility and resilience demanded by modern information systems and enterprises. It acknowledges that rigid, one-size-fits-all access permissions often fall short in today's dynamic landscape. Through risk-based access control, organizations can maintain a proactive posture, adapting their security measures in real-time to maintain a robust defense against a constantly shifting threat landscape. Risk-based access control is a valuable approach to achieving adequate adaptation even in unpredictable situations and conditions. It is a dynamic access control technique that overcomes the limits of traditional authorization approaches

by adapting access control decisions according to the level of risk associated with access requests.

Related Work

Most access control models do not satisfy the necessary adaptability required by dynamic contexts. This section presents related work proposing risk-based access control solutions with a special focus on healthcare scenarios, which is one of the application domains under study in the following sections. Recently, Atlam et al. [8] presented a novel Neuro-Fuzzy System, which is a combination of an artificial neural network and a fuzzy logic system, to estimate risk in a risk-based access control model and applied it to a children's hospital. Risk is estimated according to the action and the sensitivity of the data involved, while the patient's health status condition is not considered in the risk factor evaluations. In [29], the authors proposed a context-sensitive risk-based access control framework that grants or denies access according to the severity of the context. Doctors are bound to a permission set based on the severity of the situation, symptoms, and treatments. Risk is estimated through the correlations between requested data and the corresponding permission profile. Access is granted when the level of risk falls below a threshold (depending on the situation), but there is no ML model to support it. Another risk-based adaptive security solution for healthcare is presented in [1] where the authors refer to risk, estimated through game theory, in terms of the possibility of compromising a medical device when adapting security methods and mechanisms. In these last two proposals, the authors do not explain how the risk is quantitatively estimated and do not present any implementation to demonstrate the applicability and results of their theories. Existing solutions tend to consider access requests to patient data without mentioning other relevant scenarios for healthcare, such as doctors who have to update treatments or a nurse who has to open a medical fridge to administer insulin to a diabetic patient.

In Chapter 6, we propose and implement a risk-based access control framework that enables fine-grained access to data and resources and avoids the limits of previous models. Unlike other research proposals which only discuss qualitatively the risk estimation, we detail how the risk can be estimated through FL models. Another paper [107] adopts different ML-based approaches to dynamically decide whether or not a request should be granted or denied in a hospital management system. The best results were obtained

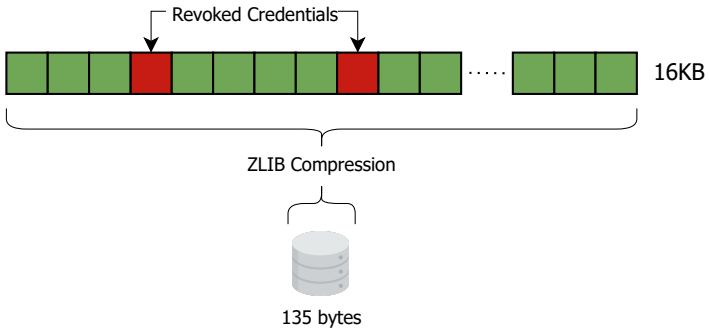


Figure 3.1: Revocation List 2020.

by combining auto-encoder for feature selection and random forest for classification. Although the proposed risk-adaptive access control model presents some common features to our solution, it mainly focuses on the authenticity and trust of the requester. It does not take into account the current health status of the patient, representing a key element of our proposal. In [66], the authors presented a fuzzy modeling technique to estimate the risk of each access request. However, the paper does not provide information on how to estimate risk and the construction of fuzzy rules requires prior knowledge of various environmental scenarios.

3.3.2 Authorization Revocation

Over time, permissions to access services, resources, or data may change. Therefore, efficiently revoking authorization rights is a crucial concern that demands proper attention. For example, an employee who has changed company should be immediately removed from the authorized users who can access internal resources. When using the access control models described above, revocation is usually achieved by modifying the enforced access control policies.

However, access could be also granted by embracing alternative approaches. Recently, we have been witnessing the widespread adoption of VCs. A user or an object can prove the ownership of a VC and be authorized to use facilities or gain access to certain resources. How to efficiently revoke VCs is currently an open challenge since the W3C has not defined any standard yet. There is only one specification proposed, which is neither on the W3C standards track. This specification, known as "Revocation List 2020" [93],

revolves around a bitstring-based mechanism for managing revocations. In this arrangement, each bit within the bitstring corresponds to an index stored in the VCs. When the bit is set to 1 the VC is revoked, otherwise it is still valid. It is worth noting that VCs are similar to PKI certificates, thus existing revocation mechanisms can easily extended to VCs. potentially they may be revoked using the same approaches such as certificate revocation list (CRL) [32] and online status protocol (OCSP) [96]. The former consists of a list of revoked VCs, while in the latter the verifier sends a request to the issuer to check the validity of the presented VC. It is worth noting that such approaches are more user-oriented since they assume large amounts of memory and reliable connections.

Therefore, there is a need for an efficient revocation mechanism in constrained environments, such as IoT, where entities may have limited storage and computing capabilities. In Chapter 6, we propose an efficient revocation mechanism for IoT networks that leverages an ECC-based accumulator. The presented revocation scheme outperforms existing approaches

Related Work

In this section, we conducted a review of the existing literature related to revoking VCs. Additionally, we also thoroughly explored works that address the revocation of certificates or credentials, which can be eventually extended to VC, with a particular emphasis on IoT networks that are central in the Cloud-to-Thing Continuum.

Verifiable Credential Revocation. Most existing studies lack a specific emphasis on constrained environments and tend to adopt traditional revocation mechanisms [28]. For instance, Abraham et al. [2] proposed to revoke VCs by including them in a revocation list. Additionally, to allow offline verification, network nodes (i.e., issuers and users) can generate attestations for valid VCs. These attestations enable verifiers to determine the freshness of the presented credential, ensuring its validity. Recently, Fotiu et al. [44] introduced a lightweight revocation mechanism for VCs tailored for IoT environments. Their design draws inspiration from the recent W3C draft [93]. The proposed approach involves maintaining a revocation list, where each VC is assigned a position. The position information is embedded within each VC. If the corresponding position is set to 1, the VC is revoked. It is worth noting that this mechanism is most suitable for scenarios with a restricted number of VCs. A similar approach is also presented in [45], wherein in a group

of IoT devices, each member is allowed to participate through a VC. The revocation list is not referenced through any information embedded in VCs, while it is included in a TXT DNS record a retrieved is identified through DNS resolution. This solution specifically refers to group member revocation and may not be suitable for use cases where devices are not clustered in groups.

Accumulator-based Approaches. Cryptographic accumulators have long been recognized as an efficient solution for revoking anonymous credentials and PKI certificates [19, 49]. However, most of the existing literature has primarily focused on user-centric revocation rather than efficient revocation in constrained networks. For example, in the protocol recently proposed by Parameswarath et al. [90], the accumulator is used to revoke user policy. The unique characteristics of accumulators, such as their constant memory size and verification time, harness them to design novel revocation schemes specifically tailored for IoT networks [25, 116]. In this direction, the most relevant related work is V'CER [60], an efficient certificate validation in constrained networks. V'CER adopts SMTs, a form of a hash-based accumulator, for performing revocation. The proposed approach achieves remarkable storage efficiency, requiring less than 3KB per node to store 1 million certificates, while also supporting offline updates.

DLT-based Approaches. Numerous studies [61, 103, 126] have highlighted the potential of DLTs in revolutionizing PKI-based authentication and authorization schemes. One particular area where blockchain technology has shown promise is in the management of certificates, with initial applications focusing on storing certificate updates and revocation information [39]. However, implementing a blockchain-based revocation system for IoT devices presents several challenges, including long latency, high energy consumption, and low transaction throughput. To address these challenges, innovative approaches [113, 117] leveraging DAG-based ledgers have emerged as a viable alternative for IoT networks. DAG-based ledgers offer a lightweight solution that maintains the same level of security as traditional blockchains while addressing the specific requirements of IoT environments.

4 Threat Modeling

According to security-by-design principles, to design and implement secure systems, security has to be considered from the very early stages. However, these principles are often neglected, while security operations are performed once the system has been realized. This chapter introduces threat modeling strategies targeting edge computing and cyber-physical systems, which are key technologies within the Cloud-to-Thing Continuum ecosystem. The proposed approaches identify threats, security controls, and MTD techniques in accordance with the features of the asset involved as well as the capabilities of data stored/transmitted.

4.1 Edge Computing Systems

Edge computing enables computation to occur on the outskirts of a system, close to data sources. Here, edge nodes locally handle a range of computing tasks, including processing, storage, caching, and load balancing of data traveling to and from the cloud [102]. Cloud services are reserved for resource-intensive tasks such as big data analytics and machine learning training. This approach is particularly well-suited for latency-sensitive applications [101], such as streaming services, gaming platforms, industrial automation, autonomous vehicles, energy platform monitoring, and home automation. In such cases, the delay caused by data traversing back and forth between devices and remote cloud data centers can adversely affect user experiences, system functionality, and even safety.

Large-scale data transmission necessitates high network bandwidth, which can be prohibitively costly. Moreover, it raises security and privacy concerns since sensitive or proprietary data is transported and processed remotely, potentially conflicting with data localization regulations.

In this complex landscape, identifying the specific threats that affect a particular deployment is a primary challenge. Risk profiles vary significantly based on individual use

cases and application domains, heavily contingent on the involved components and technologies. Security-related activities, including threat modeling, must keep pace with rapid development and deployment processes, necessitating simplification and automation where possible.

To tackle this issue, we have introduced an automated threat modeling approach for edge computing systems [21]. This approach relies on a comprehensive system model that describes the key architectural assets and associated data flows, with a specific focus on properties that could influence the applicability of threats and the corresponding countermeasures.

4.1.1 System and Data Modeling

Ecosystems, where IoT, edge, and cloud converge to form a seamless computing continuum, comprise several heterogeneous components with distinct security needs and varying computing and storage capabilities. A typical Cloud-to-Thing Continuum system consists of three layers, namely the cloud service layer, the edge layer, and the device layer. The edge layer has enough computational power to manage IoT devices and run containerized applications. Therefore, this distributed computing paradigm promotes a robust integration between cloud services and IoT. However, a drawback is the huge amount of connected devices and interactions with the edge computing layer, which considerably broadens the attack surface.

To simplify security analysis in edge computing deployments and assist developers in the phases of threat modeling and countermeasure selection, we introduce a system modeling approach. This approach empowers developers to specify key aspects related to the system's architecture, including its principal hardware and software assets, and the associated data flow that directly impacts security. this model demands minimal effort from developers to outline the essential characteristics of a system, thereby enabling the automatic generation of a threat model tailored to the specific system deployment. Figure 4.1 illustrates the asset and data modeling classification.

Asset Modeling

As previously mentioned, in an edge computing system, there are various types of assets, which belong to different architectural and functional levels. These assets fall into

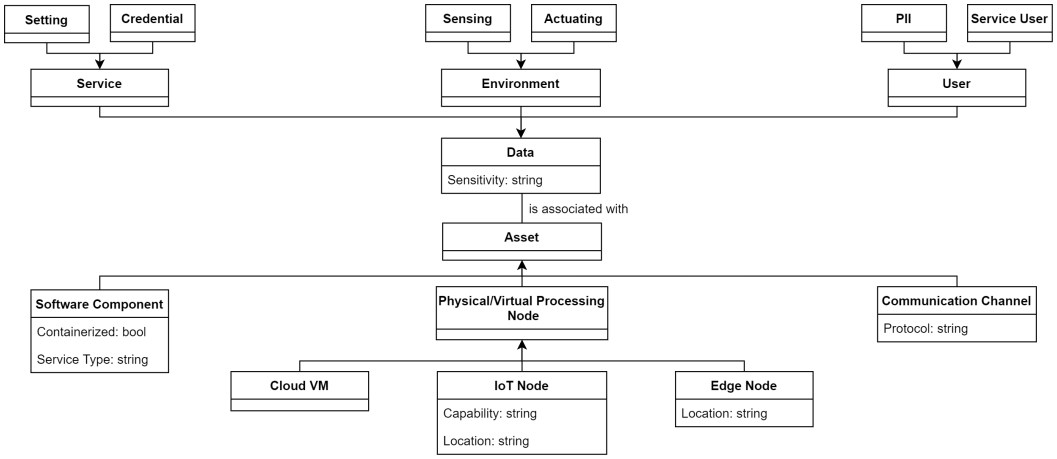


Figure 4.1: Asset and data modeling.

three primary categories: physical/virtual processing nodes, software components/modules, and communication channels.

Processing nodes are critical to different layers within an edge computing system and are responsible for executing application programs and services that embody the system's core logic. At the IoT and edge layers, these nodes a represented through physical devices, while at the cloud layer, they are virtual machines conforming to the Infrastructure as a Service (IaaS) model. We classify a *Processing node* into three distinct categories: `Edge node`, `IoT node`, and `Cloud VM`. Notably, processing nodes exhibit diverse storage and computational capabilities, affecting the potential implementation of security measures. Cloud-based resources and edge nodes generally support traditional security mechanisms, while IoT devices often accommodate only lightweight protocols due to their limited computing and storage capacities. Moreover, specific physical characteristics, such as battery-powered operation, can introduce unique threats not applicable to AC-powered nodes (e.g., battery exhaustion). To categorize IoT devices based on their processing/storage capabilities and power supply, we utilize a `capability` attribute. According to the classification proposed in [99], devices fall into categories like `Constrained`, `Limited`, `Restricted`, and `Normal` based on factors such as RAM, ROM, and power source. Constrained devices, characterized by being battery-powered, up to 10KB RAM, and up to 128KB ROM, do not support any security mechanisms. Limited devices, also battery-powered, possess RAM in the range of 10-32KB RAM, and ROM between 128 and 512KB, enabling them to support some symmetric key-based protocols. Moving up the

scale, restricted devices, which may be either battery or AC-powered, feature RAM ranging from 32-128KB and ROM spanning 512KB to 10MB. These devices exhibit greater capability, capable of implementing symmetric protocols and lightweight asymmetric key-based protocols. At the highest tier, normal devices, characterized by AC power, RAM equal to or exceeding 128KB, and ROM of 10MB or more, are the most powerful and can seamlessly support a wide range of traditional security protocols. In an edge computing context, the physical deployment location of a node also impacts its security profile, potentially exposing it to distinct threats. To account for this, we introduce the `location` attribute, with values `Protected`, accessible only by authorized personnel, and `Open`, more vulnerable due to unrestricted access).

Communication Channels are the channels established among the nodes, employing various communication protocols (e.g., Zigbee, Bluetooth, Wi-Fi). These channels can be exploited by malicious attackers to compromise the system. To address threats related to communication channels, we employ the `protocol` attribute, representing the actual communication protocol in use.

Software Component represents the third asset category. It encompasses the software components, modules, and services that contribute to implementing the system's core functionality. To simplify characterization, we employ the `service type` attribute, with values like `web-based service` (HTTP-based), `storage service` (for remote data storage), and `IoT service` (non-HTTP protocols). It is worth noting that container-based virtualization is becoming prevalent in edge computing, offering portability and rapid deployment. However, it introduces unique security challenges. To account for this, we include a `containerized` attribute, a boolean value indicating whether the software component uses container technology.

Data Modeling

To effectively model the data within an edge computing deployment from a security perspective, we established a data classification based on their subject or relevance. It is essential to note that irrespective of the data source or responsibility, they can be categorized based on their sensitivity. According to ISO27001 standards, organizations should conduct an information classification process to evaluate the data they manage and determine the necessary level of protection. To address this vital aspect, we introduced a `sensitiv-`

`ity` attribute applicable to all data types, with three potential values: `Public`, `Internal`, and `Confidential`.

Public data (e.g., public temperatures, and traffic conditions) are accessible to everyone. Internal data (e.g., user profiles, and configuration settings) can only be accessed by specific system entities. Confidential" data (e.g., credentials). Based on the sensitivity level, the unauthorized disclosure, alteration, or destruction of data would result in a low, moderate, or significant level of risk, respectively. Let us now illustrate the considered data classification.

User-related data comprises information related to the end-users, including data used for system interaction (credentials, profiling information, etc.), referred to as `Service user data`. Additionally, it comprises personally identifiable information (*PII*), which uniquely identifies individuals. GDPR has compelled organizations to safeguard PII securely. Article 9[1] of the European GDPR specifies the types of personal data that require explicit consent for processing. This category includes information revealing racial or ethnic origin, political opinions, religious beliefs, trade union membership, biometric data for identification, health or sexual orientation data, judicial records, electronic communication (IP addresses), geographic location, etc.

Environmental data refers to data exchanged/stored in an edge system, which not only concerns users but also the system's operational environment. Environmental data can either originate from sensors or be used for controlling actuators. We categorize environmental data into `Sensing` and `Actuating` data, with their sensitivity varying based on the application domain.

Service data encompasses all other data not directly related to end-users or the environment. It includes service credentials, configuration parameters (e.g., deployment details), and additional information used by services and applications, regardless of their criticality.

4.1.2 Our Methodology

The presented approach offers an automated method for performing threat modeling of edge computing systems, removing the need for specialized security expertise. Develop-

[1] https://www.privacy-regulation.eu/en/article-9-processing-of-special-categories-of-personal-data-GDPR.htm

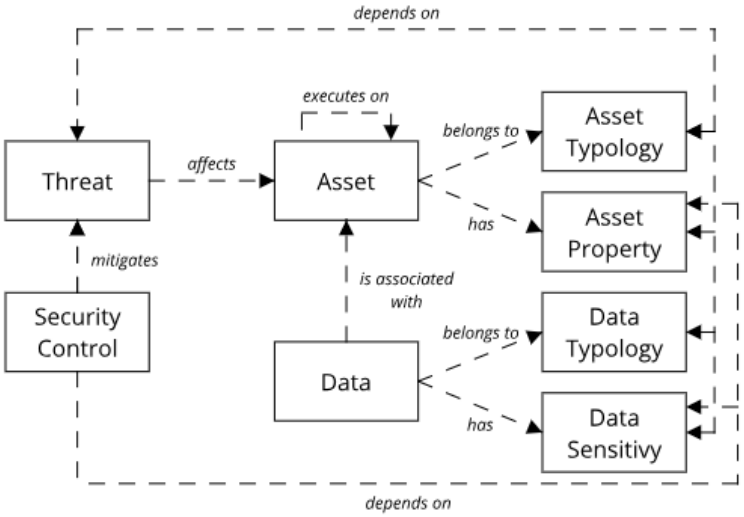


Figure 4.2: Security data model.

ers are only required to define the system under examination by identifying and labeling assets and data with the appropriate attributes, as outlined in our earlier system model. Following this modeling phase, they will be automatically provided with a comprehensive threat model for the system, along with a set of security controls to implement. This automation is facilitated by the *security data model* depicted in Figure 4.7, which effectively connects asset and data classification, characterization, and the corresponding threats and security controls.

The security data model is implemented through a *Threat Catalogue*, which encompasses a wide array of threats categorized according to the STRIDE Threat Model [77] (spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege). In addition to providing a detailed threat model for a specific edge deployment, our objective is to automatically associate threats with appropriate mitigation measures. To achieve this, we identified a set of security controls from the NIST Security Control Framework [82] for each threat, serving as mitigation measures. The NIST framework comprises over 900 distinct security controls, spanning 18 control families, including both fundamental controls and control enhancements, which strengthen the core security capability of a control.

According to our approach, a subset of applicable security controls is selected, based on the actual computational power and storage capacity available on each node. This

#	Asset Typology	Threat	STRIDE Category	Asset Properties	Data Typology	Data Sensitivity	Security Controls
1	Edge Node	Camouflage	Spoofing	Location:*Open*	-	-	IA-3, IA-3(1, 3), IA-5, ...
2	Edge Node	Hardware Trojan	Tampering	Location:*Open*	-	-	SI-3(3), SI-16, ...
3	Edge Node	Unauthorized Access Control	Elevation of Privilege	-	PII, User Service, Credential, Setting	Internal, Confidential	AC-17, AC-17(1, 2, 4, 5, 6), ...
4	IoT Node	Battery Draining	Denial of Service	Location:*Open*, Capability:*Constrained, Limited, Restricted*	-	-	PE-2, PE-3, ...
5	IoT Node	Denial of Service	Denial of Service	-	-	-	SC-5
6	IoT Node	Exhaustion of Power	Denial of Service, Spoofing	Capability:*Constrained, Limited, Restricted*	-	-	PE-11
7	Communication Channel	Jamming	Denial of Service, Spoofing	-	-	-	IA-3, SC-5, ...
8	Communication Channel	Network Key Sniffing	Information Disclosure	Protocol:*Zigbee*	Credential	Confidential	SC-8, SC-13, ...
9	Communication Channel	Message Elimination	Information Disclosure, Spoofing, Tempering	-	-	Internal, Confidential	AC-17, SA-18, ...

Figure 4.3: Extract of the threat catalogue.

information is explicitly defined for processing nodes through the `capability` attribute. For software components, it depends on the capability specified for the processing node designated for their execution (which must be established by the developer during system modeling). In Figure 4.3, we present a segment of the Threat Catalogue. Notably, if an asset property does not impact a threat, it is simply omitted from the table. If a threat is unaffected by a field in the catalogue, that field is filled with "-". The last column of the table lists some of the NIST controls that serve as effective countermeasures for thwarting or mitigating each threat. As mentioned earlier, during the countermeasure selection phase, a subset of these controls will be chosen based on the capabilities of the involved assets.

The Threat Catalogue has been meticulously curated by aggregating threats from various sources. To date, we have gathered more than 150 threats specific to cloud services, web-based applications, storage services, IoT and edge devices, and network protocols, drawing from established standards and scientific studies [4, 31, 51, 89, 108, 111, 125]. While the effectiveness of our approach relies on the comprehensiveness of the catalogue, it can be readily expanded to incorporate new threats and address emerging challenges.

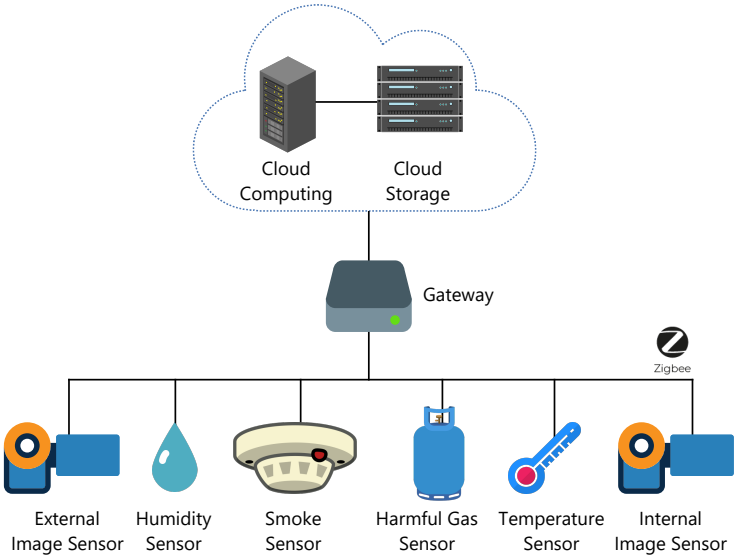


Figure 4.4: Smart home security monitoring system CPS-based case study.

4.1.3 Case Study

In this section, we present an example of how our proposed approach is applied. The deployment under study was introduced in [110] and sketched in Figure 4.4. The smart home comprises various IoT devices interconnected with an edge node, which possesses the capability to locally manage the devices and execute software services. We decided to employ Microsoft's Threat Modeling Tool for our case study, which was enriched with the assets and data flows detailed in Section 4.1.1, as well as their associated threats.

Following our approach, we need to determine the properties and types of each asset within the system, as well as the types and sensitivity levels of data that are either stored or transmitted. For the sake of conciseness, our focus will be primarily on edge nodes, IoT devices, and their communication channels, while other component types will be excluded from consideration. The edge node serves as a gateway situated within the residence, making its location classified as "protected" since only the owners have physical access to it. Concerning the end devices involved, our system comprises smart bulbs, smart locks, smoke detectors, smart TVs, and smart cameras. In Figure 4.5, we present the specifications of the gateway, IoT devices, and one instance of a communication channel. All IoT devices are connected to the gateway via this communication channel asset. We as-

Asset	Asset Properties	Data Typology	Data Sensitivity
Gateway	Location:*Protected*	-	-
TV	Location:*Protected*, Capability:*Normal*	Service User	Confidential
Internal Camera	Location:*Protected*, Capability:*Normal*	Sensing, PII	Internal, Confidential
External Camera	Location:*Open*, Capability:*Normal*	Sensing, Actuating	Public
Smoke Detector	Location:*Protected*, Capability:*Restricted*	Sensing, Actuating	Confidential
Internal Lock	Location:*Protected*, Capability:*Constrained*	Sensing, Actuating	Internal
External Lock	Location:*Open*, Capability:*Constrained*	Sensing, Actuating	Confidential
Bulb	Location:*Protected*, Capability:*Limited*	Sensing, Actuating	Public
Comm. Channel Ext Lock-Edge	Protocol:*Zigbee*	Sensing, Actuating, Credential	Confidential

Figure 4.5: Asset specifications.

sumed that smart bulbs support the ZigBee protocol, smart locks, and smoke detectors utilize BLE, while smart TVs and cameras are compatible with the Wi-Fi protocol.

Once we have successfully modeled the system, we proceed to identify potential threats and select appropriate countermeasures. The threats and their corresponding mitigation strategies are automatically retrieved from the Threat Catalogue. As mentioned earlier, each threat depends on the asset to which it pertains (in terms of its type and properties) and the nature and sensitivity of the data involved. For instance, referring to Table 4.3, the gateway is not susceptible to threats applicable to nodes located in public areas, but it is vulnerable to the *Camouflage* threat. All IoT devices are susceptible to *Battery Draining*, and except for smart cameras and smart TVs, they are also susceptible to the *Exhaustion of Power* threat. Communication channels, regardless of the protocol employed, are consistently susceptible to *Jamming* attacks. Moreover, communication channels that transmit data classified as internal and/or confidential are at risk of the *Message Elimination* threat. Lastly, due to the adopted protocol, the communication channel linking smart bulbs to the edge node is exposed to the *Network Key Sniffing* threat.

4.2 Cyber-Physical Systems

Cyber-physical systems are composed of computation and physical processes/systems. In a CPS, the physical layer senses environmental data, which is then sent to the cyber layer. The cyber layer comprises computing resources that orchestrate control and monitoring of the physical world through feedback loops. Here, information processing is exploited to adjust system parameters and/or make changes to physical processes. Technology advancements as well as the increasing availability of sensors and actuators are paving the way to CPSs across many sectors such as smart manufacturing, intelligent transportation, personalized health care, emergency response, and electric power generation and delivery. Application domains can be very different, often entailing a large number of heterogeneous components that complicate the task of CPSs with a certain degree of fidelity [34].

However, despite the differences among CPS architectures, there are some core elements in common. CPSs benefit from the convergence of cloud, edge, and IoT into the Cloud-to-Thing Continuum paradigm, which is the enabling technology for providing interconnectivity and intelligence to these systems. From an architectural and technological standpoint, CPSs can be depicted using a layer-based model. In this model, each asset is associated with a layer cloud, edge, and IoT [20] based on its computation capabilities and technology.

It is worth noting that, the huge number of connections and interactions among the layers considerably broadens the attack surface. Attackers may exploit both cyber vulnerabilities and physical weaknesses, leading to substantial damage to physical assets, compromising operational safety, and negatively impacting product quality and performance. Given this landscape, CPS design should integrate security from the beginning, by identifying essential security measures based on existing threats and enforcing them directly into the system architecture. Unfortunately, identifying the actual threats that affect a given deployment is a challenging task. Risk profiles vary significantly based on individual use cases, application domains, components, and technologies involved.

Additionally, enforcing security measures does not prevent an attacker from succeeding in compromising an asset. Therefore, CPSs both need secure and resilient mechanisms, especially when referring to critical contexts. According to NIST [86], *resilience* (or resiliency) is *"the ability to anticipate, withstand, recover from, and adapt to adverse condi-*

tions, stresses, attacks, or compromises on systems that use or are enabled by cyber resources". As outlined in a recent NIST publication (SP160 Volume 2 Rev1 - Developing Cyber-Resilient Systems: A Systems Security Engineering Approach) [87], cyber-resilience measures may have different goals, such as preventing/avoiding the successful execution of an attack, minimizing the degradation/interruption of delivered services, limiting current and preventing future damages, restoring compromised functionalities, and/or restructuring/modifying systems or sub-systems to reduce risks. Therefore, modern CPSs demand a high level of adaptability to different circumstances. Implementing resilience within such systems calls for a variety of approaches, including dynamic reconfiguration, resource allocation, obfuscation, deception, diversity, and the introduction of temporal or contextual unpredictability. These strategies collectively fall under the umbrella of what is known as MTD techniques.

MTD [59] represents a proactive security approach that revolves around continually altering a system's configuration to increase the uncertainty for the attacker and reduce the attack success probability. MTD strategies not only enhance system security but also provide it with resilience capabilities. They empower systems to *anticipate* attackers' actions by preemptively shifting the attack surface, thwarting reconnaissance efforts, and to *adapting* to adverse conditions, *preserve system availability* and *recover from attacks* by dynamically reconfiguring the system at various levels, leveraging diversity and redundancy techniques.

To increase the security and resilience of CPSs, we extended the work presented in [21]. The proposed approach [22] is a model-based design methodology that embraces the principles of security-by-design and the strategic application of MTD techniques. Our methodology leverages an extensive system model, capable of describing the key architectural components and their associated data flow, with a focus on the specific properties that may impact the applicability of threats and of associated countermeasures and related MTD strategies. To the best of our knowledge, we are the first to propose a model-based MTD approach tailored for CPS.

4.2.1 System and Data Modeling

To model CPSs, we relied upon the system and data modeling proposed in [21]. Our layered model, encompassing cloud, edge, and IoT, finds applicability across various CSP-

based scenarios. To illustrate, we delve into its relevance in the context of Industry 4.0 where CPSs play an increasingly crucial role. Traditionally, enterprise architectures adhere to the Purdue Reference Model [120]. This model effectively delineates the interconnections and interdependencies of all the main components within a typical Industrial Control System (ICS) by categorizing ICS architecture into two distinct zones: IT and OT. The Purdue Model consists of the following levels:

- **Level 0 - Physical Process**: Encompasses physical components such as sensors and actuators responsible for product assembly.
- **Level 1 - Control**: Comprises systems overseeing and issuing commands to Level 0 devices. This level encompasses Programmable Logic Controllers (PLCs), Remote Terminal Units (RTUs), and Intelligent Electronic Devices (IEDs).
- **Level 2 - Supervisory**: Consists of supervisory systems that manage overall processes, including interfaces like human-machine interfaces (HMIs) and engineering workstations.
- **Level 3 - Manufacturing Operations Systems**: Encompasses systems supporting the orchestration of production workflows, including batch management and data historians.
- **Level 4/5 - Enterprise**: Corresponds to the enterprise network, serving as the repository for data from ICS systems, which informs business decisions.

The Purdue model's strength lies in its structured hierarchy, offering clarity regarding the functionality of each component within smart manufacturing by assigning it to a specific level. However, modern control system networks often blur the boundaries between these levels. Additionally, the demand for real-time OT data, coupled with the integration of cloud, edge systems, and backend services, is paving the way for more flexible system architectures. However, we claim that this model can still be useful to understand how information flows and to identify potential attack vectors, revised versions have emerged. For instance, the European Union Cyber Security Agency (ENISA) introduced a Purdue Model variant that acknowledges a Level 3 Industrial Internet of Things (IIoT) platform communicating directly with Level 1 IIoT devices.

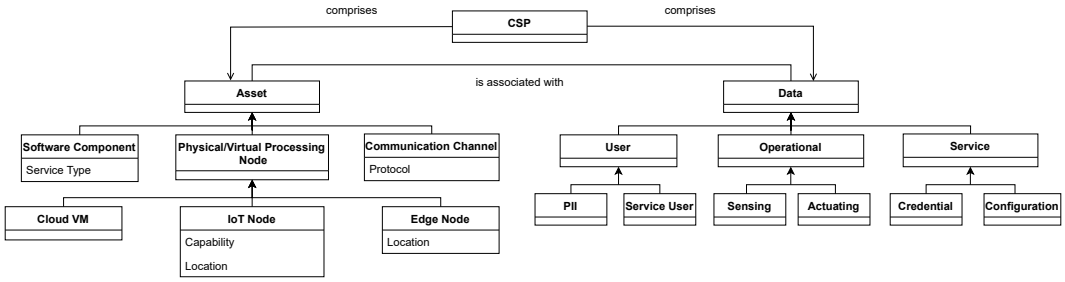


Figure 4.6: CPS - Asset and data modeling.

The adoption of a three-layer model addresses different requirements, including data management and security, while also streamlining the representation of a modern industrial system. Nevertheless, the levels of the Purdue model can be easily mapped onto our three-layer model. Specifically, Level 0 which mainly comprises sensors and actuators corresponds to the device layer. Levels 1 through 3, which comprise elements physically deployed within the manufacturing implant and aim at managing and controlling devices and/or communication with the IT network, are associated with the edge layer. Lastly, the IT zone is bound to the cloud layer. For visual clarity, Figure 4.6 depicts the extended asset and data modeling for CPS.

4.2.2 Our Methodology

The proposed methodology enables to carrying out of a guided threat modeling process of the system under study, which enables spotting existing security issues according to the types and features of the assets and concerning information. This method also allows for determining security countermeasures to implement to prevent or mitigate identified threats. Furthermore, since attackers may still be able to defeat implemented cybersecurity safeguards, our methodology helps to identify appropriate moving target defense techniques to improve the overall system resilience.

Our approach leverages both the *system model*, built by identifying the involved assets along with related properties and the information flow, and a reference *security data model*, sketched in Figure 4.7, which suitably links together the concepts related to assets, asset properties, threats, security controls, and MTD techniques.

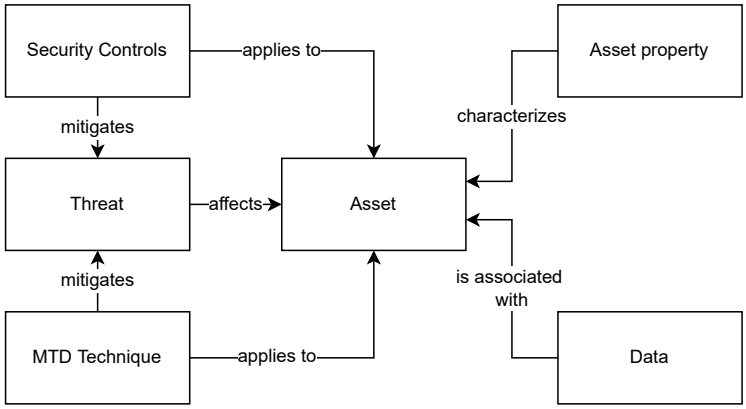


Figure 4.7: Security data model for CPS.

Threat Modeling and Countermeasure Selection

As outlined by the model in Figure 4.7, the threat modeling process is facilitated by directly linking threats to assets and applying filters based on their properties and on involved data flow, which directly influences their applicability. We denote the set of threats as $\mathcal{T}$ and the set of security controls $\mathcal{SC}$. We define a function, denoted as F, that models the applicability of threats to assets. Given an asset $a \in \mathcal{A}$, along with the associated properties $P \in \mathcal{P}$, and the data $D \in \mathcal{D}$ being either stored or transmitted, $F(a, P, D, \mathcal{T})$ returns the set of applicable threats $\mathcal{T}_a \subset \mathcal{T}$. Each threat $t \in \mathcal{T}_a$ if t is associated with a, P, and D.

The process of countermeasure selection entails identifying the set of technical security controls necessary to mitigate existing threats to an asset. This step is accomplished by the association between threats, assets, and standard security controls. These security controls are implemented through specific security mechanisms, which can potentially impact system performance or face resource constraints. To address this concern, asset properties play a crucial role in refining the set of applicable security controls based on the capabilities of the components involved, including computational power and storage capacity. For example, a *constrained* IoT node may lack the computational resources to execute public-key algorithms for message authenticity verification. Conversely, a *normale* IoT node possesses sufficient resources to effectively implement such countermeasures. This information is made explicit for processing nodes through the specification of the `capability` property. For software components, it depends on the capability specified for

the processing node used for their execution. Therefore, we introduce a function denoted as M, which models the feasibility of security controls for mitigating threats. Given the same input parameters as F, except that $\mathcal{T}_a$ is provided instead of $\mathcal{T}$, $M(a, P, D, \mathcal{T}a)$ returns the set of feasible security controls, denoted as $SC\mathcal{T}_a$, capable of mitigating threats within $\mathcal{T}a$. Each security control $sc \in SC\mathcal{T}_a$ is associated with a and P.

MTD Techniques Identification

Considering resilience requirements, our methodology allows identifying a set of MTD techniques. These techniques can be adopted to proactively prevent or mitigate the realization of threats, even when suitable countermeasures have already been deployed (as determined through the threat modeling process). As previously mentioned, MTD techniques harness spatial and temporal diversity, redundancy, and shuffling to dynamically alter the system's attack surface over time. From the architectural side, MTD techniques can be classified based on the level at which they work [83]:

- *Platform-level*: MTD techniques refer to computing platforms and execution environment levels, encompassing elements such as CPU architecture, instruction sets, memory systems, and the operating system.
- *Software-level*: MTD techniques revolve around application logic, architecture, or code.
- *Network-level*: MTD techniques relate to communication channels, encompassing communication and security protocols, network addresses, and topologies.

While MTD techniques are directly linked to assets and threats in our security data model, their actual applicability to a given asset hinges on available capabilities. Consequently, MTD techniques undergo filtering based on threats, assets, and asset properties. For instance, consider the *firmware reconfiguration* technique, which involves altering the entire application running on an IoT device to thwart attackers from gaining complete control. This strategy necessitates sufficient local memory on the node to store a second firmware version. In our modeling, this approach may be feasible for a normal IoT device but inapplicable to those categorized as constrained or restricted.

We denote the set of MTD techniques as $\mathcal{M}$ and introduce a function, denoted as B, which determines whether a technique can enhance the resilience of an asset. Given an

#	Asset	Threat	STRIDE	Properties	Data	Data Sensitivity	NIST Security Controls
1	Edge Node	Camouflage	Spoofing	Location:*Open*	-	-	IA-3, IA-3(1, 3), ...
2	Edge Node	Node Replication	Spoofing	Location:*Open*	-	-	IA-3, IA-3(1,3), ...
3	IoT Node	Battery Draining	Denial of Service	Location:*Open*, Capability:*Constrained, Limited, Restricted*	-	-	PE-2, PE-3, ...
4	IoT Node	Exhaustion of Power	Denial of Service, Spoofing	Capability:*Constrained, Limited, Restricted*	-	-	PE-11
5	Communication Channel	Network Key Sniffing	Information Disclosure	Protocol:*Zigbee*	Credential	Confidential	SC-8, SC-13, ...
6	Communication Channel	Message Elimination	Information Disclosure, Spoofing, Tempering	-	-	Internal, Confidential	AC-17, SA-18, ...
7	Cloud VM	Denial of Service	Denial of Service	-	-	-	SC-5, ...
8	Software Component	Reverse Engineering	Information Disclosure	ServiceType:*web-based, IoT*	-	Internal	SR-9, SR-9(1), ...

Figure 4.8: Extract of the Threat Catalogue that outlines the relationship between assets, asset properties, threats, data, and NIST security controls.

asset $a \in \mathcal{A}$, its set of properties $P \in \mathcal{P}$, and the set of applicable threats $\mathcal{T}_a$ derived from F, $B(a, P, \mathcal{T}_a)$ returns the set of pertinent MTD techniques, denoted as $\mathcal{M}_a \subset \mathcal{M}$. Each MTD technique $m \in \mathcal{M}_a$ is associated with a, P, and $t \in \mathcal{T}_a$.

Threat Catalogue

We extended the Threat Catalogue presented in Section 4.1.2, which currently includes more than 150 different threats each tailored to specific domains, including cloud services, web-based applications, storage services, IoT devices, edge nodes, and network protocols. These threats have been derived from existing standards and scientific studies such as [4, 31, 51, 89, 111, 119, 125]. Concerning security controls, this extended version of the catalogue still relies on the NIST Security Control Framework. Furthermore, the catalogue was enriched with a set of MTD techniques recently proposed in the literature [10, 23, 30, 83, 114]. These techniques can be applied at various architectural levels to enhance the resilience of CPS-based architectures. They may be applied at different architectural levels to design more resilient CPS-based architectures. The current catalogue implementation[2] relies upon the Microsoft Threat Modeling Tool that was customized with the CPS assets, as well as the related threats, countermeasure, security controls, and MTD techniques.

[2] https://github.com/ci-ma/ThreatModeling-MTD-CPS

#	Asset	Threat	Properties	MTD Level	MTD Technique
1	Edge Node	Camouflage	-	Network	Identity Virtualization
2	Edge Node	Node Replication	-	Network	Identity Virtualization
3	IoT Node	Battery Draining	Capability:*Constrained, Limited, Restricted*	Network	Honeypot
4	IoT Node	Exhaustion of Power	Capability:*Constrained, Limited, Restricted*	Software	Crypto-protocol reconfiguration
5	Communication Channel	Network Key Sniffing	Protocol:Zigbee	Network	Dynamic Re-keying
6	Communication Channel	Message Elimination	-	Network	Dynamic Re-keying
7	Cloud VM	Denial of Service	-	Platform	VM Migration
8	Software Component	Reverse Engineering	ServiceType:*web-based, IoT*	Software	Code Obfuscation

Figure 4.9: Extract of the Threat Catalogue that outlines the relationship between assets, asset properties, threats, and MTD techniques.

Figures 4.8 and 4.9 provide excerpts from the Threat Catalogue. The former encompasses threats and security controls, while the latter details MTD strategies. In these tables, asset properties that do not influence a threat or MTD technique are omitted, and fields that are irrelevant to a threat or MTD technique are marked with a hyphen ("-"). In Table 4.3, the last column enumerates some of the NIST security controls and control enhancements (presented in parentheses) that should be enforced to mitigate the corresponding threat. It is important to emphasize that the selection of security controls is contingent on the capabilities of the assets within a given deployment scenario. The effectiveness of our approach hinges upon the comprehensiveness of the catalogue, but it remains readily adaptable for the inclusion of new threats and the accommodation of emerging security challenges.

4.2.3 Case Study

This section discusses how our proposed methodology can be applied to the case study introduced in Section 4.1.3 and depicted in Figure 4.4. The system comprises several IoT devices deployed within the *perception layer*. Devices are seamlessly integrated with the control *control layer*, represented by a gateway component that manages their operations. The gateway interfaces with the *decision layer*, which harnesses cloud-based services to analyze sensory data and make informed decisions, and optimize the entire system.

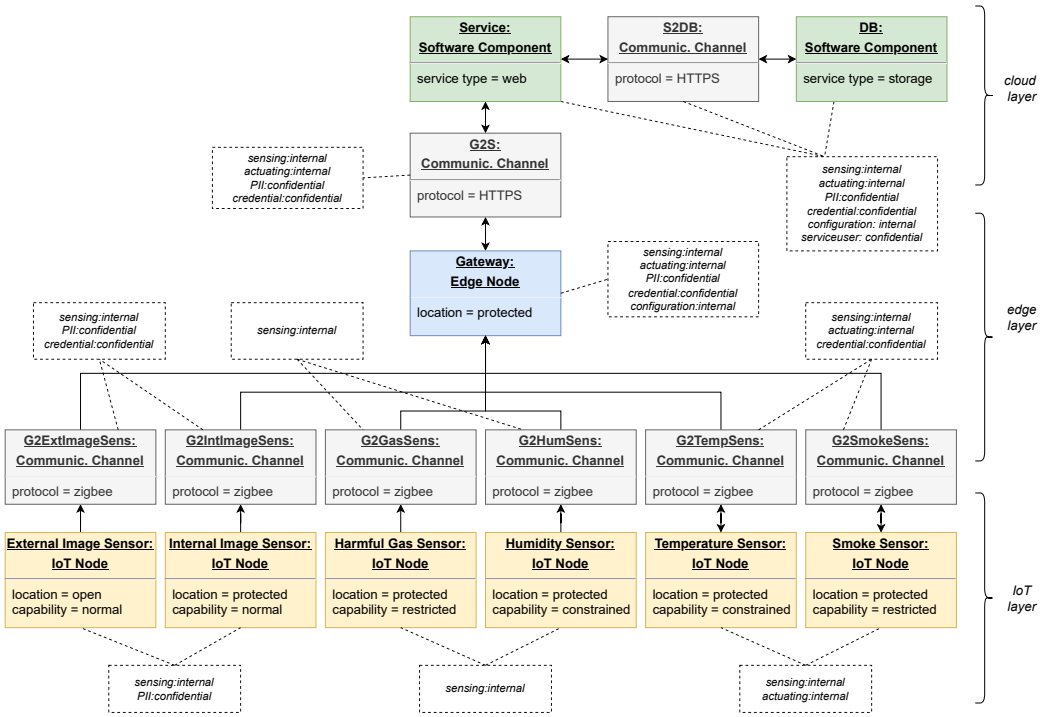


Figure 4.10: Smart home security monitoring system CPS-based case study - system modeling

At first glance, it becomes evident that the layers within this system align closely with those in our three-layer model. The correlations between the perception and device layer, as well as between the decision and cloud layer, are straightforward to establish. Furthermore, the gateway within the closed-loop system, which accepts control instructions from the decision layer, can be mapped onto the edge layer. In line with the graphical representation in Figure 4.10, we have adopted the system modeling approach discussed in Section 4.2.2 to depict our case study application. Specifically, we have identified and categorized the assets, which are illustrated as rectangular solid boxes, distinguished by different colors based on their sub-types (yellow for IoT Nodes, light blue for Edge Nodes, light gray for Communication Channels, and green for Software Components). Additionally, we have specified relevant properties for each asset, denoted as name-value attributes within each box. Furthermore, we have identified and categorized the data and have associated them with the corresponding assets, indicating their sensitivity levels. These data are represented as dashed boxes, linked to the assets.

In the architectural diagram, we depicted the gateway node as an asset categorized under the "Edge Node" type. Its `location` property was set to *Protected* to indicate that the device is deployed within a habitation, and only the house owners have physical access to it.

Within the device layer, we represented the image sensors, smoke sensor, temperature sensor, humidity sensor, and harmful gas sensor, which constitute the home security monitoring system, as assets falling under the IoT Node type. Additionally, the cloud-based web application and the underlying database service were categorized under the Software Component type.

Furthermore, we modeled all communication channels connecting the device and edge layers as Communication Channel node types. For these channels, we specified the ZigBee `protocol` as a property due to its widespread usage, which stems from its excellent network coverage, real-time data transmission capabilities, and cost-effectiveness. We also designated HTTPS as the `protocol` used for communications between the gateway node and the upper layer, as well as between the two cloud-based services. A more graphical representation of the considered assets, data, and properties is shown in Table 4.5.

In this work, our focus is primarily on threat modeling and MTD approaches for edge nodes and IoT devices, without delving into further analysis for other component types. For each asset a within the home security system monitoring $\langle \mathcal{A}, \mathcal{D} \rangle$, we need to determine its properties $P \in \mathcal{P}$ and the type and sensitivity of data $D \in \mathcal{D}$ that it stores and/or transmits. Once we modeled the CPS-based architecture, we proceeded with identifying threats and selecting countermeasures. Additionally, we explore MTD strategies, all of which are drawn from our knowledge base, codifying security expertise. Each threat t is associated with an asset a based on its type, properties P, and the relevant data D. Security controls and MTD techniques to mitigate these threats are identified according to the assets and their properties.

For example, referring to the extract from the catalogue in Table 4.3, the gateway is not susceptible to threats like *Camouflage* or *Node Replication*, which typically affect nodes with an *Open* location. However, it may be vulnerable to *Outage*, a threat that affects edge nodes exposed to unauthorized access. As for the data involved, since the gateway handles all types of data due to its centralized nature, we use the symbol "-" to denote this. With regards to IoT devices, except for image sensors, they are susceptible to *Battery Draining* and *Exhaustion of Power* threats. Independently from the adopted

protocol, all communication channels that transmit *Internal* and/or *Confidential* data are consistently exposed to the *Message Elimination* threat. Furthermore, in the current deployment, devices and the gateway communicate using the ZigBee protocol, making all network communications involving *Credential* data (sensitivity level: "Confidential") vulnerable to *Network Key Sniffing*. After identifying the threats to the CPS-based architecture under analysis, those responsible for designing a secure and resilient system must ensure the proper implementation of corresponding security controls and MTD techniques. For instance, the *Exhaustion of Power* threat affects the smoke sensor. To mitigate this threat, the *P-11 EMERGENCY POWER* security controls should be enforced. This control entails providing "*an uninterruptible power supply to facilitate an orderly shutdown of the system and/or transition of the system to long-term alternate power in the event of a primary power source loss*". Additionally, to enhance the device's resilience, considering its energy limitations, the recommended MTD technique is *Crypto-protocol Reconfiguration*, which is the least expensive in terms of energy consumption compared to other strategies.

5 Adaptable Access Control

Adaptable access control represents a paradigm shift in the management of accessing data, resources, and services. Traditional approaches are static, being unable to face the complex and dynamic Cloud-to-Thing Continuum ecosystem. Indeed, the huge amount of evolving entities that interact, demand flexible, context-aware, and responsive mechanisms that can adapt to the ever-changing conditions of the digital world. This chapter explores adaptable access control from different perspectives. First, we discuss how access control policies can be efficiently disseminated across other areas all over the world. Then, we present a risk-based access control framework for healthcare, where access to resources is granted according to context information including the patient's health status. Finally, we introduce a dynamic approach to managed authorization permissions to join a collaborative ML training.

5.1 Access Control Policy Distribution

Access control policies specify how access is managed and who may access services, resources, and information under what circumstances. The way a policy is structured is fundamental to enable adaptability. However, due to the high dynamicity of modern systems, policies can be modified or replaced. Thus, their effective dissemination is another crucial concern.

For example, multi-region applications must behave as a single service that undergoes the same access control policies with a unique coherent view of involved data. In this context, data management is one of the major challenges. On the one hand, centralizing information in a single repository can jeopardize system robustness and hamper performance due to the constant need for data transfers. On the other hand, replicating data across multiple regions introduces the risk of synchronization issues.

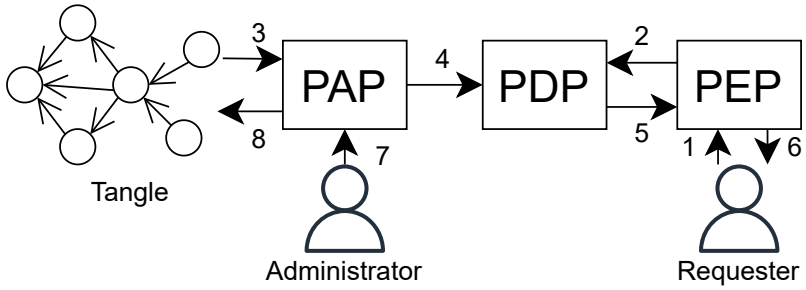


Figure 5.1: Proposed access control architecture.

In the pursuit of high availability, multi-region cloud platforms adopt different strategies for replicating data across various geographic locations, spanning even international borders. Consequently, these platforms must ensure that all data replicas, used by different instances of applications and services, adhere to the same access control policies. However, maintaining a cohesive view of access control policies across distinct regions proves to be a formidable task. When a policy is updated in one region, these modifications must swiftly propagate to all other instances of applications and services. Hence, the choice of technology for achieving this synchronization becomes a key consideration in this context [56].

To overcome such concerns, DLTs are considered promising solutions to share access control policies in multi-region applications where involved services may be spread across different countries. In this direction, we evaluate the feasibility of employing a DAG-based ledger (the IOTA Tangle) to store access control policies [74] by implementing a prototype version of an access control architecture and comparing it with a corresponding implementation that uses a document-oriented NoSQL database.

5.1.1 Reference Architecture

We consider an access control architecture, sketched in Figure 5.1, which can be deployed in multi-region applications and comprises the components of XACML architecture [109]. In the following, we highlight the responsibility of each component.

- *Policy Enforcement Point* (PEP): It intercepts access requests and constructs a query in a language comprehensible to the PDP for each request. Subsequently, it makes

access decisions to grant or deny access to a resource or service based on the outcomes of policy evaluations conducted by the PDP.

- *Policy Decision Point* (PDP): It assesses access requests according to the relevant access control policy.
- *Policy Administration Point* (PAP): Administrators utilize this component to load, update, and revoke policies. It serves as the source of policies for the PDP.
- *Tangle*: A distributed ledger that ensures a consistent perspective of access control policies across multiple regions.

We suppose that requests have already been authenticated by an identity provider. In the following, we detail the authorization flow highlighted in Figure 5.1.

1. The requester sends an access request to the PEP.
2. The PEP extracts information, such as user attributes and the resource to be accessed, from the incoming request. It then formulates a query that the PDP can understand.
3. The PAP retrieves policies from the Tangle.
4. The collected policies are supplied to the PDP, which assesses the request using the associated policy queries.
5. The PDP provides the policy decision back to the PEP;
6. The PEP either grants or denies the access request based on the outcome provided by the PDP.
7. An administrator, using the PAP, can introduce new policies, update existing ones, or remove those that are no longer needed.
8. The PAP interacts with the Tangle to fulfill administrator requests.

As we discuss multi-region applications, it is important to note that these components, with the exception of the Tangle, are duplicated across various instances. Thanks to the

Tangle's characteristics, we can ensure that regardless of where an application is deployed, it undergoes the same access control policies. This consistency arises from all instances of the PAP accessing policies from a singular source.

5.1.2 Policy Properties

The technology adopted to share policies has to preserve integrity, authenticity, and confidentiality. In this section, we discuss how the Tangle addresses such properties.

Integrity

A policy should be stored in its original form to prevent potential data breaches and unauthorized access. The Tangle is designed to ensure data immutability, ensuring that any information stored cannot be altered.

Authenticity

Using policies that have been created or altered by malicious individuals can lead to undesirable consequences. To address this concern, we employed a private Tangle. As a result, only legitimate users have the right to access the Tangle for policy management. However, this choice may still come with certain limitations, primarily pertaining to the challenge of monitoring Tangle access. One potential solution is to accept policies exclusively from a predefined set of addresses. Nevertheless, given our decision to employ zero-cost transactions in the IOTA network, the sender and receiver addresses are not considered. To overcome this challenge, we should distribute IOTA tokens to entities authorized to interact with the Tangle. Another alternative is to employ external mechanisms such as digital signatures or HMAC (Hash-based Message Authentication Code).

In this study, we use a private Tangle accessible solely to authorized users, leaving the exploration of alternative techniques for future research.

Confidentiality

In certain scenarios, maintaining the confidentiality of policies becomes highly significant. For instance, a patient might have reservations about disclosing the fact that a doctor with a specific specialization is accessing their medical records, as it could compromise

the privacy of the patient. In this research, we operate under the assumption that a private Tangle remains inaccessible to inquisitive or malicious individuals. Nevertheless, ensuring confidentiality can be effectively achieved by implementing encryption techniques, such as those found in IOTA's Masked Authenticated Messaging (MAM) library, which enables the secure transmission of encrypted data to the Tangle.

5.1.3 Implementation

We implemented the first prototype of the architecture sketched in Figure 5.1. The PEP was realized through Envoy[1], which is an L7 proxy and communication bus specifically designed for handling large-scale modern services.

For our PDP, we employed Open Policy Agent (OPA) [2], a lightweight general-purpose policy engine service that decouples policy evaluation from their enforcement. Policies are defined using Rego[3], a declarative policy language supported by OPA, and stored on the Tangle.

In our implementation, we opted for a private IOTA network consisting of a Coordinator, a Spammer, and an IOTA Node, with multiple instances of this latter. The Spammer periodically sends messages to the Tangle to maintain a minimal message load, thereby facilitating transaction approval in accordance with the IOTA protocol.

The PAP plays a central role in all the interactions with the Tangle. It serves as an abstraction layer for both the mechanism responsible for enabling zero-cost transactions for policy loading, updating, and revocation, as well as the *Bundle Service*, a software application that aggregates Rego policies from the Tangle and makes them available to OPA. To store policies on IOTA, we employ zero-cost transactions embedding the policy text as part of the transaction data. However, due to the immutability of the DLT, when a policy needs to be modified or revoked, new transactions must be sent to the Tangle. These transactions include references indicating the type of operation they represent.

To update policies in OPA without requiring a restart, OPA can periodically download a bundle of policies for immediate enforcement. The Bundle Services assembles this bundle, taking into account the order of transactions. Additionally, the service extracts policies corresponding to the requested bundle, ensuring that only rules belonging to the

[1] https://www.envoyproxy.io/docs/envoy/latest
[2] https://www.openpolicyagent.org
[3] https://www.openpolicyagent.org/docs/latest/policy- language

specific bundle are examined. All essential information, including the *package name*, *type of action*, and *policy text* is embedded in the message of transactions as shown in Listing 1.

```
{
   "package": <policy_package>,
   "action": <add | delete>,
   "policy": <policy>
}
```

Listing 1: Message structure.

On the flip side, information about the bundle to which a policy belongs is obtained through the transaction tag. IOTA provides users with the capability to search for transactions based on their tags, making it easy to gather policies associated with a specific bundle. While the mechanisms we propose could potentially be replicated on other DLTs, IOTA offers several features that render it particularly suitable for managing access control policies. Our decision is primarily driven by the following considerations:

1. IOTA offers the ability to associate a tag with a transaction, streamlining the process of searching for policies within the Tangle. Not all DLTs provide this feature, so an alternative would involve embedding a tag within the text of a transaction, which would increase the time required to locate the desired policies.

2. Thanks to zero-cost transactions in IOTA, we can neglect the cryptocurrency aspect. These transactions significantly simplify the complexities involved in adding or revoking policies. Otherwise, we would need to establish a cost-value for each transaction, ensuring that administrators and bundle services have sufficient funds to carry out their tasks.

3. Finally, the Tangle structure enables reducing the time to memorize a new transaction. Transactions just created are memorized on the Tangle, while in blockchain-based solutions a transaction will not be memorized until it is stored into a block.

It is important to note that, given our focus on multi-region applications, the components of our architecture, with the exception of the Coordinator and the Spammer, will be replicated across different geographical areas.

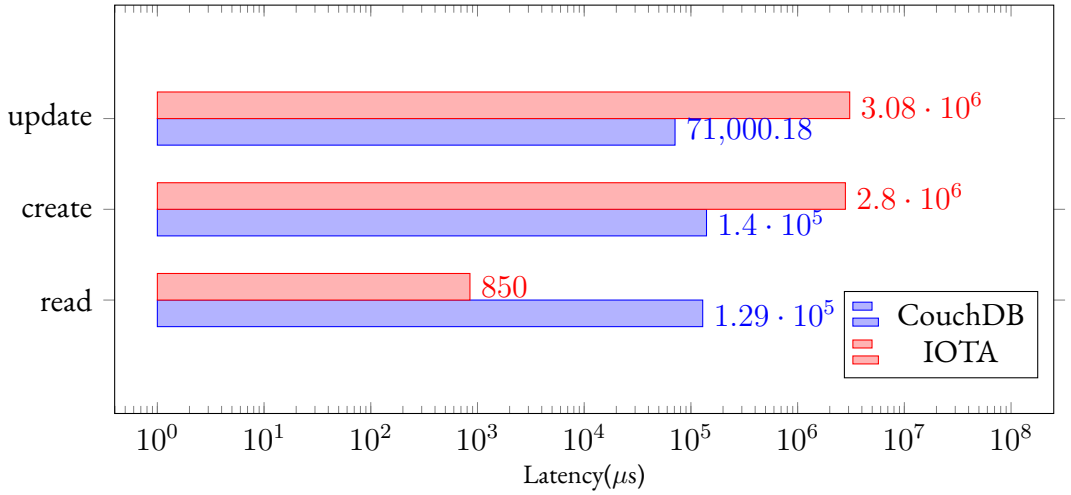


Figure 5.2: Comparison of operation performance on a logarithmic scale.

5.1.4 Evaluation and Results

In order to evaluate the feasibility of using a Tangle to distribute access control policies, we have conducted some preliminary experiments aiming to evaluate the performance of the Tangle under different circumstances. In this regard, we compared our proposal with the same architecture replacing the Tangle with CouchDB[4], a document-oriented NoSQL database that exploits a Multi-Version Concurrency Control (MVCC) protocol to enable the synchronization and replication of documents over one or more instances.

Operation Performance

We evaluated the latency between the Bundle Service and data sources (IOTA and CouchDB) while creating, reading, and updating policies under diverse load conditions. Both configurations were subjected to testing by initiating a progressively increasing number of requests in 60 seconds. Specifically, we simulated the actions of 10 virtual users, each starting with a delay of 5 seconds. For the purpose of evaluating all operations related to CouchDB and reading from IOTA, we maintained a constant throughput of 1000 requests per minute. However, for creating and updating operations on IOTA, we reduced the throughput to 100 requests per minute. Figure displays the average latency, measured in microseconds, for the aforementioned operations.

[4]https://couchdb.apache.org

By observing the graph, we can conclude that IOTA outperforms CouchDB in terms of reading policies. Conversely, CouchDB offers a more consistent latency, with all operations taking roughly 100 milliseconds. Although IOTA is approximately one order of magnitude slower than CouchDB when it comes to creating and updating policies, this duration can still be deemed acceptable, considering that these operations are not frequent. We anticipated a longer duration for updating policies due to the immutability of the ledger, involving two transactions: one for revoking the policy and another for creating a new version.

Additionally, we assessed the latency between the creation of a new policy and its accessibility. In this context, IOTA exhibits an average latency of approximately 600 milliseconds, while a new policy becomes available after roughly 150 milliseconds when utilizing CouchDB. It's important to note that the time required by IOTA in this case is significantly slower than that observed for creating and updating policies as presented in Figure 5.1.4. This discrepancy arises because, in the evaluation of availability latency, we did not subject the IOTA network to the same stress as in the previous experiments. In the case of CouchDB, there are no notable variations compared to the earlier experiments.

5.2 Risk-based Access Control in Healthcare

Adaptable access control mechanisms are particularly relevant in certain scenarios such as healthcare, which is characterized by continuously evolving conditions. Recently, the medical industry has been remarkably reshaped by recent technological advancements. For example, the widespread IoMT is revolutionizing healthcare, providing reliable and on-demand services for patients and the elderly through lightweight sensors [123]. These sensors remotely monitor patient health and collect vital physiological data [50], which are helpful for emergency medical decisions and chronic disease detection.

Caregivers must be able to offer assistance at any time and in any place, highlighting the need for flexible access control models that can adapt to various situations. For instance, in emergencies like car accidents or when the condition of a patient suddenly worsens and the patient is not already hospitalized, electronic medical records become crucial for saving lives. This means granting access to emergency information for ambulance personnel or even passersby caregivers, despite their lack of permission in normal circumstances

[122]. Relying on static security policies and mechanisms to prevent data access or resource use can impede life-saving efforts. Therefore, the trade-off between privacy and safety is a fundamental aspect to consider while designing and developing authorization solutions.

We consider risk-based access control as a valuable strategy for ensuring effective adaptation, even in unpredictable situations and settings. In healthcare, assessing risk involves considering factors such as the requester's context and the patient's health status. Estimating the risk tied to a patient's condition is a significant challenge, given the need to correlate a vast amount of vital physiological data.

ML holds promise for analyzing large datasets and predicting patient health, supporting medical operations. However, challenges like data heterogeneity, incompleteness, timeliness, and privacy constraints, driven by regulations such as the European GDPR, hinder traditional centralized ML approaches [55]. In addition, relying only on the amount of data collected by one medical organization may be insufficient to train adequately an ML model. This is why we advocate for FL as a promising approach to leverage patient data while preserving their privacy [3]. FL allows the training of a global model in a central server without the need to aggregate data in the server itself. Data are kept locally in the institutions where they originated, thus preserving privacy and ownership. Moreover, we propose to enrich FL with a blockchain system to avoid weaknesses related to a centralized server, such as single point of failure, low scalability, and to limit possible biases inducing to prefer some partial models over others [33].

To address the aforementioned challenges, we presented a Federated Learning Risk-Based Authorization Middleware for Healthcare (FRAMH) [72]. that allows users to access patient data and equipment according to the context information of the requester and patient including the health status. We leverage FL to train an ML model that assesses the current patient's risk level. All partial models are securely stored in the blockchain ensuring their integrity and trustworthiness. To the best of our knowledge, we are the first to adopt FL to infer risk related to the patient's health status and in general to estimate health status risks in risk-based access control models for healthcare scenarios [**atlam2020risk**]. In the following sections, we primarily delve into the proposed risk-based access control model.

5.2.1 Access Control Model

The FRAMH access control middleware has been developed to meet the adaptability requirements of contemporary healthcare situations. It utilizes two kinds of visibility to guide access control choices, taking into account the balance required between security and safety considerations: context information related to the requester and the patient to control access to patient data and the health status risk. In subsequent sections, we will outline the underlying risk and context models in FRAMH, along with the essential support services necessary to implement context-aware, risk-based access control decisions.

Risk Classification

FRAMH categorizes patient health status severity into three levels based on the risk classification introduced in [29], which identifies patient conditions as either *critical*, *serious*, or *stable*. While a more granular risk classification is possible, it would complicate policy management, especially when dealing with overlapping policies for different health risk levels. Notably, we have not encountered practical scenarios necessitating finer-grained risk classification. However, FRAMH can easily adapt to different risk classifications with minimal adjustments if necessary. Figure 5.3 illustrates the adopted risk levels and their relationship with safety and privacy considerations. As the situation becomes riskier, safety takes precedence over privacy. Conversely, in stable situations, privacy is prioritized over safety.

- *Critical*: Indicates a critical threat to the patient's life. Privacy becomes secondary and requests for data or medical equipment access are approved. Medical professionals must obtain information promptly to maximize efforts to save the patient's life.
- *Serious*: Patient conditions are considered urgent. In such a situation, doctors can access patient data from other departments. We grant access to riskier data to expedite patient treatments.
- *Stable*: Users can only access authorized data. There is no reason to grant access to additional information since the patient is not facing a life-threatening situation.

This risk classification is used to label the predictions generated by our ML model.

Figure 5.3: Considered risk levels.

Context Modeling

The context model is used for representing, storing, and retrieving context information, which is crucial for constructing access control policies and making access control decisions. To balance simplicity in policy specification with a high degree of flexibility, we employ the context model illustrated in Figure 5.4. This context model comprises elements that encompass all relevant context information and their attributes, which govern access control policies. The *Context Provider* component includes context information to determine access decisions for both the *Requester* and the *Patient*. The requester typically represents a healthcare professional, but in some cases, it could be a relative or friend. For instance, in a smart home environment, access to medication stored in a smart medical refrigerator, which regulates access to critical drugs like insulin, might need to be granted in emergencies where the patient cannot access the medicine themselves within a specified timeframe [62].

The context provider comprises context elements, each characterized by a set of attributes. Both the requester and patients possess a `physical context`, which includes `time` and `location` context information. The former can be either an `instant` or an `interval`, while the location attribute may refer to an `absolute` or `relative` geographical position (i.e., longitude and latitude or room within a hospital). Time-related information allows for access regulation during specific periods, while geographical data can grant access based on the requester's current location.

The requester's context includes `role`, `department`, and `usual`. The role indicates the healthcare professional category or the relationship between the patient and the requester, such as being a family member or friend. The department contains information related to the requester's specialization. `Usual` refers to whether the requester has previously visited the patient. Patient-specific context information covers `treatment` and `health status` (critical, serious, or stable).

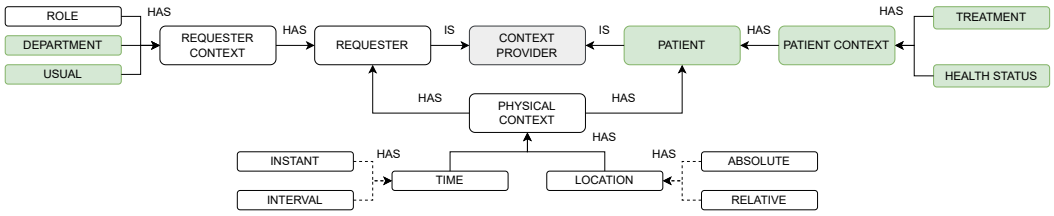


Figure 5.4: The context model employed to build risk-based access control policies.

Our model aligns with context information characteristics based on an ontology, like the Web Ontology Language (OWL) [75]. The domain-independent part of the context ontology (in white boxes) consists of reusable, common information applicable across various domains. In the domain-specific segment (colored boxes), we incorporate key concepts derived from the Health Level Seven (HL7) Reference Information Model [38].

Access Control Model Formalization

Concerning the context model shown in Figure 5.4, it classifies information into two categories: requester attributes, which belong to the requester context, and patient attributes, which belong to the patient context. However, due to the continuously evolving nature of health status, we classify it as general context information, similar to time and location. Our formal model relies on the following entities:

- *Requester*: we denote with $\mathcal{U}$ the set of requester u;
- *Requester Attribute*: we denote with $\mathcal{RA}$ the set of possible values that a requester attribute ra can assume;
- *Patient Attribute*: we denote with $\mathcal{PA}$ the set of possible values that a patient attribute pa can assume;
- *Resource*: we denote with $\mathcal{R}$ the set of resources r that can be accessed by a requester u;
- *Action*: we denote with $\mathcal{A}$ the set of actions that a requester u can perform on a resource r associated to patient p;

Table 5.1: Examples of Risk-based Context-Sensitive Policies.

Requester Attribute	Patient Attribute	Resource	Action	Context Information
Role: "Doctor", Hospital: "X" Department: "Orthopedic"	Treatment: "Orthopedic" Hospital: "X"	Data: Neurological	Read: False, Delete: False	HealthStatus: "Stable"
Role: "Doctor", Hospital: "X" Department: "Orthopedic"	Treatment: "Orthopedic" Hospital: "X"	Data: Neurological	Read: True, Delete: False	RequesterLocation *is equal to* PatientLocation, HealthStatus: "Serious"
Role: "Nurse", Hospital: "X" Department: "Diabetes"	Treatment: "Diabetes" Hospital: "X"	Equipment: Refrigerator	Open: True, Close: True, IncreaseTemperature: True, DecreaseTemperature: True	RequesterLocation *is equal to* PatientLocation, HealthStatus: "Stable"

- *Context Parameter*: we denote with $\mathcal{CP}$ the set of possible values that a context parameter cp^i, with $i \in \mathcal{U} \cup p$, can assume. A context parameter is characterized by a name cpn and value cpv, thus, cp is tuple $\langle cpn, cpv \rangle$.

The set of context parameter names $\mathcal{CPN}$ is determined by pre-specified parameter names. In this paper, we consider as context information $\mathcal{CPN}$ = {`Time`, `Location`, `HealthStatus`}. Context is defined as follows:

Definition 1 (Context). *A Context C is a set of n context parameters $cp \in \mathcal{CP}$. For each cp_i^k, cp_j^z, with $i \neq j$ and $k = z$, we have that $cp_i^k.name \neq cp_j^z.name$. However, with $i \neq j$ and $k \neq z$, it is possible that $cp_i^k.name = cp_j^z.name$. In a context C, there cannot be two cp belonging to the same requester u with the same name, although this may occur for a requester and a patient p.*

Context information has a primary role in dynamically granting or denying access to resources. In risk-based access control policies, such information enables adapting to different circumstances overcoming the limits of rigid static access control approaches designed to always apply the same access control policies that do not consider environmental changes and unpredictable situations. We formally define an access control policy as follows:

Definition 2 (Access Control Policy). *An access control policy ap is a set of clauses given by $RA \cup PA \cup A \cup R \cup C$ with $RA \subset \mathcal{RA}$, $PA \subset \mathcal{PA}$, $A \subset \mathcal{A}$, and $R \subset \mathcal{R}$.*

A requester u with attributes RA can perform the actions A, under the context C, on the patient resources PR whose owner has attributes PA.

For clarity, in Table 5.1, we report some examples of risk-based context-sensitive policies, such as the first access control policy claims an orthopedic of hospital X cannot read neurological data of a patient whose conditions are stable. The adopted mechanism allows many-to-many access control since policies are not bound to a specific requester or patient.

5.3 Trustworthy and Dynamic Participation to Federated Learning

FL is an ML technique that fits within the Cloud-to-Thing Continuum paradigm. Indeed, it operates at the edge of the continuum, where data is processed on individual devices or nodes, and model updates are sent back and forth to a parameter server, which is usually hosted in the cloud. Thus, it adheres to the principles of edge computing and is a way to leverage the computational capabilities of edge devices.

While the FL paradigm addresses certain AI-related challenges, such as privacy preservation, it still faces security vulnerabilities, including potential attacks like model poisoning and inference attacks [106]. In terms of trustworthiness, the primary obstacle hindering FL adoption in third-party applications is the presence of malicious clients and servers, which can adversely affect FL training performance and introduce malicious backdoors [64]. Moreover, the conventional FL architecture, based on the client-server model, suffers from several drawbacks, including a single point of failure, limited scalability, and vulnerabilities to tampering with the global model. This can result in potential biases favoring certain partial models over others [85].

To address such challenges, we propose a blockchain-based trustworthy federated learning as a service (TruFLaaS) [73] framework. Our solution provides trustworthiness among 3rd-party contributors to the FL process by leveraging blockchain, smart contracts, and decentralized oracle networks (DONs). Specifically, we employ a dynamic validation strategy to aggregate partial models, enhancing the overall quality of the global model. Clients' models are validated by a smart contract through a sample of the validation dataset given by the service provider through a DON. By evaluating the partial models on defined quality metrics (e.g., accuracy), we can generate high-quality global models. We associate a level of trust, updated during each round, with each client in order to properly weigh

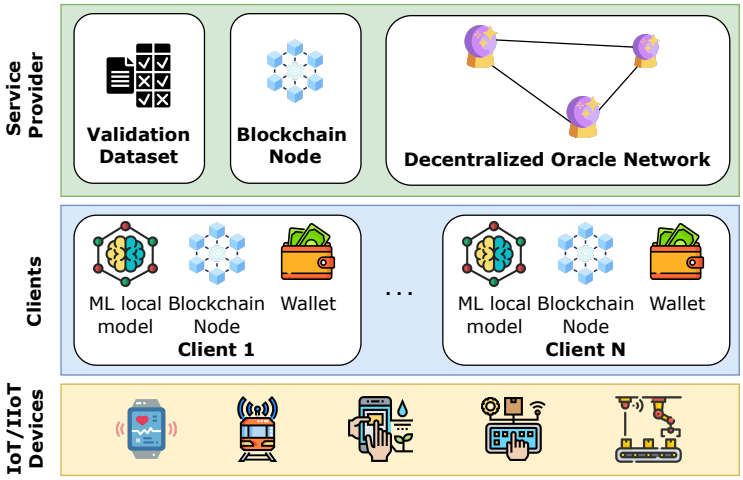


Figure 5.5: TruFLaaS architecture.

their contributions. Thus, the trust level dynamically affects participants' contributions. Furthermore, when the trust level drops a given threshold, the VC of that client, which regulates its participation, is revoked. Therefore, our aggregation strategy adapts to the trust level of each client.

5.3.1 TruFLaaS Architecture

FL is becoming increasingly valuable for building machine learning models in a decentralized manner without compromising data privacy. Despite its advantages, the setup and utilization of FL can be both costly and time-intensive, especially in certain sectors like industry and healthcare, where the necessary infrastructure and expertise are often lacking. FLaaS offers clients a convenient way to employ FL with minimal additional effort and technical expertise, enabling them to offload the demanding tasks of algorithm development and optimization. Furthermore, FLaaS is adaptable to accommodate diverse participant requirements during FL training. In this section, we introduce the architecture of TruFLaaS, which is illustrated in Figure 5.5.

Service Provider

The service provider is the entity that provides FLaaS to clients for tasks beneficial for all participants. For example, smart manufacturing enterprises using specific machinery

often share a common objective of preventing breakdowns. However, even though this goal is shared, clients may have varying requirements concerning metrics, the number of participating nodes, and aggregation strategies. For instance, one client might prioritize obtaining the global model as quickly as possible, setting a threshold for the number of participants needed to aggregate the collected partial models. Conversely, another client may be less concerned about obtaining the global model within a short time frame, instead preferring to wait to accumulate a greater number of contributions. Consequently, the service provider must offer a flexible service capable of accommodating these diverse demands.

Clients furnish this information to the service provider, who utilizes it to create a smart contract tailored to the FL process in compliance with the client's specifications. Additionally, the service provider is responsible for client registration and issuing valid identifiers for blockchain interaction. This ensures that only identifiable and authorized clients participate in the FL training process.

Moreover, as the service is provided directly by the service provider for a task under its control (e.g., predictive maintenance of its machines), it can be assumed that the service provider possesses a sufficiently large validation dataset to assess partial models effectively, as referenced in [67]. After each round of FL training, the service provider incorporates a sample of this dataset into the blockchain. These data are used to validate all partial models before aggregation. To prevent potential model manipulation attacks, it's essential that the validation sample differs between distinct rounds. Otherwise, malicious participants could exploit it to create manipulated partial models that pass validation checks, jeopardizing the integrity and effectiveness of the global model. Consequently, for secure and dependable data injection into the blockchain, the service provider must rely on a DON capable of accurately fetching off-chain data and delivering it to the blockchain.

Oracles function as trusted third-party entities that act as intermediaries between blockchains and external systems. They enable smart contracts to perform computations using real-world inputs and outputs. An oracle is essentially a software component that interacts with, validates, and verifies external data sources before relaying that information to the blockchain. TruFLaaS relies on oracles to supply the validation dataset to the smart contract. These oracles are responsible for providing the smart contract with the validation dataset managed by the service provider. However, relying on a single oracle introduces a central point of failure, potentially conflicting with the decentralized principles

of blockchain technology and posing security risks. This issue, commonly known as the blockchain oracle problem [16], has prompted the emergence of DONs as a solution [14]. A DON utilizes a combination of multiple independent oracle node operators and various reliable data sources to provide decentralized and secure access to off-chain information. By leveraging the collective intelligence of multiple independent nodes, a DON helps to ensure the reliability and accuracy of data inputs into the blockchain network, while also maintaining the decentralization and security that blockchain promises.

In TruFLaaS, we employ a DON as an intermediary layer between the service provider and the blockchain. Without a DON, validation data would need to be directly published on the blockchain. Unfortunately, this could lead to an unintended consequence where all participants could potentially exploit the publicly available validation data to create a partial model that, even if malicious, could pass the validation phase. To safeguard the integrity and security of the FL process, validation data should only be provided after meeting the necessary requirements for aggregation. For instance, in the context of predictive maintenance, a malicious client might attempt to construct a partial model that performs well on the validation data while failing to accurately estimate the remaining useful lifetime (RUL) when it falls below a certain threshold.

Client

Clients gather data from IoT and IIoT devices that are utilized to train local models, regardless of the specific ML algorithm used. After completing the training process, the client transmits the partial model to a smart contract deployed on the blockchain. Consequently, each client has a module for executing FL-related tasks and hosts a node of the blockchain to join the service. Before joining an FL, clients must express their intention to participate in such a process. If the existing procedures do not meet their requirements, clients have the option to present new conditions to the service provider to establish a fresh FL training setup that meets their needs.

Blockchain Node

As previously discussed, we utilize blockchain technology and smart contracts to enhance trustworthiness among unfamiliar participants in the FL process. Consequently, the service provider is responsible for deploying blockchain nodes. Clients have the option to

either locally run a blockchain node or connect to one of the nodes deployed by the service provider.

On one hand, operating a blockchain node offers clients direct visibility into FL processes. However, it comes with the potential drawback of consuming a substantial amount of client resources, which can pose a challenge for clients with limited computing power or storage capacity. Therefore, it is crucial to carefully weigh the trade-offs between direct monitoring and resource utilization when deciding whether to run a blockchain node or connect to one of the proxy nodes and submit their partial model. This flexible configuration enhances the ease of using FL, as each participant is relieved of the burden of managing a blockchain node and only needs to focus on training partial models and transmitting them to the relevant smart contract.

5.3.2 TruFLaaS Protocol

This section explores the primary stages necessary for establishing trustworthiness within a FLaaS environment. Let's consider a scenario where a service provider, denoted as s, extends FL training capabilities, represented as $f_l \in \mathcal{F}$, to its client group, denoted as $\mathcal{C}$. This service allows clients to collaboratively train a global model, denoted as mg_l, based on specified requirements for a given task.

To participate in s's FLaaS offering, a client, referred to as $c_i \in \mathcal{C}$, must have previously registered with the service provider. Once the registration process is complete, c_i obtains a DID issued by s, which grants it the authorization to engage with the FLaaS services.

Starting and Joining FL Training

A client c_i can either join an existing f_l or start a new one if the requirements implemented by existing processes do not satisfy its demands. TruFLaaS employs a robust authorization workflow, illustrated in Fig. 5.6, that leverages DIDs and VCs to regulate all interactions. In the following, we detail the steps involved in initiating a new f_l or becoming a member of an existing one:

1. c_i provides its DID and specifies the requirements that f_l must address. The parameters that can be customized are summarized in Table 5.2.

Table 5.2: Configuration Parameters.

Parameter	Value
Minimum rounds	Positive integer
Minimum participants	Positive integer
Budget	Positive integer
Aggregation algorithm	JSON
Structure of the model	JSON
Privacy techniques	JSON
Metrics	JSON

2. If there are no existing smart contracts sc_l that meet the client's requirements, s creates a new sc_l responsible for verifying and aggregating partial models based on the specified criteria. However, if such an sc_l already exists, please refer to step 3).

3. s issues a verifiable credential $vc_{i,l}$ signed with its DID, allowing c_i to interact with the deployed sc_l.

4. After completing local training, c_i signs $vc_{i,l}$ obtained previously with its DID, generating a verifiable presentation $vp_{i,l}$. It then submits $vp_{i,l}$ and the partial model $mp_{i,l}$ to sc_l.

5. sc_l validates the authenticity of $vp_{i,l}$ using s's DID, which was responsible for releasing $vc_{i,l}$, and subsequently grants or denies participation in f_l.

It is important to note that initiating a new f_l is a resource-intensive operation and should be avoided if existing training meets the client's requirements.

Trust Level

Each client, denoted as c_i and belonging to the set $\mathcal{C}$, is assigned a trust level denoted as $t_{i,l}$ within the range $[0, 1]$.

As highlighted in [13], most trust models in P2P networks differentiate trust towards a peer into two categories: direct trust and indirect trust. Direct trust is established based on prior interactions with a specific peer, while indirect trust relies on the global reputation of that peer.

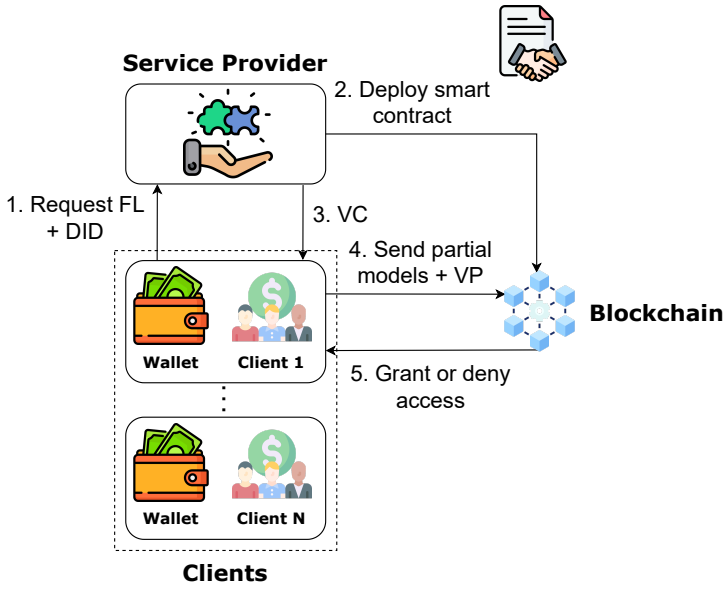


Figure 5.6: Authorization workflow.

We define the *Transaction Acceptance Rate* ($TAR_{i,l}$) as follows:

$$TAR_{i,l} = \frac{TA_{i,l}}{T_{i,l}} \tag{5.1}$$

Here, $TA_{i,l}$ represents the number of accepted transactions, and $T_{i,l}$ represents the total number of transactions made, both pertaining to the f_l process.

The *Global Trust Value* (GT_i) is defined as:

$$GT_i = \frac{\sum_{j=1}^{N} t_{i,j}}{N} \tag{5.2}$$

In this equation, trust levels are accumulated from clients who have participated in previous or ongoing FL processes. Leveraging both direct and indirect trust, we calculate the trust level as follows:

$$t_{i,l} = \frac{GT_i + TAR_{i,l}}{2} \tag{5.3}$$

These trust values are updated at each round through the specific smart contract (see Fig. 6.2, step 11). At this stage, a consideration is made regarding the revocation of $vc_{i,l}$ from client c_i if their $TAR_{i,l}$ does not meet the minimum requirements, i.e., fails to surpass a predefined threshold. In each round, the average TAR (μ_{TAR}) among all partic-

ipants and the corresponding standard deviation (σ_{TAR}) are computed. Subsequently, an estimation is made regarding whether a client, considering the number of remaining rounds, can exceed the threshold value set at $\mu_{TAR} - \sigma_{TAR}$. If it is determined that the client cannot meet this threshold, their corresponding $vc_{i,l}$ is revoked, and they are excluded from further participation in f_l.

Validation and Aggregation

After initiating a novel f_l or joining an existing one, a client is provided with the necessary $vc_{i,l}$ required for contributing to that training. Below, we elaborate on the validation and aggregation workflow depicted in Figure 6.2. These steps are repeated for each round k:

1. Each $c_{i,l} \in \mathcal{C}_l$ collects local data from the deployed IoT/IIoT devices.
2. The collected data are utilized to locally train a partial model $mp^k_{i,l}$.
3. Each $c_{i,l}$ provides $mp^k_{i,l}$ and $vp_{i,l}$ (obtained by signing $vc_{i,l}$ through its DID) to sc_l. sc_l validates and aggregates all partial models MP^k_l. Algorithm 1 outlines the algorithm implemented by the smart contract for this purpose.
4. sc_l forwards $vp_{i,l}$ to a smart contract responsible for participant authorization.
5. The authorization smart contract will grant or deny access to f_l. Specifically, it jointly verifies the validity of $vp_{i,l}$ and ensures that the embedded $vc_{i,l}$ has not been revoked.
6. Before aggregating all partial models MP^k_l, sc_l waits until the aggregation requirements are met and validates MP^k_l against a validation set V^k_l, a subset of the complete validation set $\mathcal{V}_l$ provided by a DON d.
7. d requests V^k_l from s;
8. s provides V^k_l to d. Importantly, for two distinct rounds j and z, where $j < z$, V^z_l must not be equal to V^j_l to prevent a client $c_{i,l}$ from crafting mp^z_i to perform well on a known V_l.
9. d returns V^k_l to sc_l.

10. sc_l validates MP_l^k against V_l^k. To be accepted, a partial model must meet or exceed a specific performance threshold. The Interquartile Range (IQR) method is used to detect outliers. This method is robust to extremely large or small values and is employed to calculate the threshold as $Q1 - 1.5 * IQR$, where $Q1$ is the first quartile of the data.

11. Based on the collected metrics, sc_l sends the updated reputations to a smart contract responsible for tracking the trust levels (t_i) of each c_i.

12. The smart contracts calculate the trust levels T^k for each $c_{i,l}$ and return them to sc_l.

13. sc_l aggregates all validated $mp_i^k \in MP_l^k$, weighting them according to the corresponding $t_i^k \in T^k$.

14. The resulting global model mg_l^k is distributed to all $c_{i,l}$ participants.

It is important to emphasize that transparent collaboration among smart contracts plays a vital role in ensuring a trustworthy FLaaS ecosystem. This design choice guarantees:

- All MP_l^k are validated and aggregated without biases, ensuring the correctness of gm_l^k.
- Only $c_{i,l}$ that satisfy the process requirements can participate in f_l.
- Reputations of $c_{i,l}$ are calculated by a smart contract using the output from sc_l. Thus, we ensure the correctness of t_i for each participant.

Incentive and Penalization

Additionally, incentives and penalization mechanisms are in place. To initiate a new f_l or join an existing one, clients use tokens as an incentive mechanism. Tokens are purchased from s and earned through positive participation. Tokens are also required to participate in an established f_l to discourage malicious behavior. Participants providing incorrect

Algorithm 1 Smart Contract - Validation & Aggregation

Input: $dids^k, partialModels^k, valSet^k, metric$
Output: $globalModel^k$
$acceptedPartialModels^k \leftarrow []$
$acceptedDids^k \leftarrow []$
$performances^k \leftarrow []$
for $i \leftarrow 1, partialModels^k.size()$ **do**
    $mp \leftarrow partialModels(i)^k$
    $performance \leftarrow evaluate(mp, valSet^k, metric)$
    $performances^k.push(performance)$
end for
$threshold \leftarrow generatedThreshold(performances)$
for $i \leftarrow 1, dids^k.size()$ **do**
    $did \leftarrow dids(i)^k$
    $mp \leftarrow partialModels(i)^k$
    $accepted \leftarrow false$
    if $performances^k(i) \geq threshold$ **then**
        $acceptedPartialModels^k.push(mp)$
        $acceptedDids^k.push(did)$
        $accepted \leftarrow true$
    end if
    $updateReputation(did, accepted)$
end for
$t^k \leftarrow getTrustLevels(acceptedDids^k)$
$globalModel^k \leftarrow \frac{\sum_{i=1}^{M} t(i)^k acceptedPartialModels(i)^k}{\sum_{i=1}^{M} t(i)^k}$

contributions may have a portion of their tokens withdrawn in proportion to their contribution quality score (GT_i). This ensures that all participants have a vested interest in contributing high-quality work and helps maintain the integrity of the f_l.

When creating a new f_l, a budget (b_l) in tokens is locked into the corresponding sc_l by the initiating $c_{i,l}$. This budget serves as an incentive for participation in newly started

5.3.3 TruFLaaS Evaluation

To validate our proposal and compare it with the existing literature, we focus on two significant use cases: predictive maintenance and the detection of botnet attacks. These use cases hold substantial relevance in industrial deployment scenarios and necessitate the gathering of data from diverse distributed sources.

Dataset

For the predictive maintenance use case, we employed the NASA Turbofan Jet Engine dataset [97]. This dataset is widely recognized and serves as a foundational benchmark in NASA's engine degradation modeling. It facilitates the estimation of RUL for the engines under consideration. The dataset was generated through simulations of a commercial modular aero-propulsion system and is composed of four sub-datasets. These sub-datasets contain temporal readings from 21 sensors, including variables such as temperature and fuel flow ratio. Each sub-dataset represents different combinations of operational conditions and fault modes.

To make effective use of this dataset, we initiated a data pre-processing step, which involved eliminating features with inconsistent values. Additionally, since the training set did not provide RUL values directly but only the number of time cycles an engine had been in operation, we had to calculate RUL manually. For the purposes of our subsequent evaluation, we assumed that RUL decreases linearly over time, reaching a value of 0 at the end of the engine's lifecycle. For each engine, we computed RUL as the difference between the maximum time cycle and the current time cycle: RUL = $max\_time\_cycle - time\_cycle$. Moreover, in line with standard practice for regression problems, we normalized the input values. Finally, we partitioned the dataset into training and testing subsets, simulating the behavior of an FL network during the training phase using data from 100 engines.

Regarding the botnet attack detection use case, we employed the N-BaIoT dataset [76], which consists of genuine network traffic data collected from nine commercial IoT devices that were genuinely infected by the Mirai and BASHLITE malware strains. This dataset encompasses various types of malicious traffic, including network scanning and firmware attacks. As a result, it supports multi-class classification, with ten attack classes and one 'benign' class. In our data preparation process, we initially used a Label Encoder to convert the target values for each sample into numerical representations. These target values signify the type of network traffic, either benign or one of the ten potential attacks. Subsequently, we applied one-hot encoding to these values, transforming them into vectors for each sample. Additionally, we normalized each feature using a MinMaxScaler, scaling the data within the range of [0,1]. To reduce the feature dimensionality, we implemented a feature selection mechanism based on an ExtraTreesClassifier. Tree estimators were employed to calculate feature importance based on impurity calculations. This information was then used in conjunction with the SelectFromModel meta-transformer to eliminate irrelevant features.

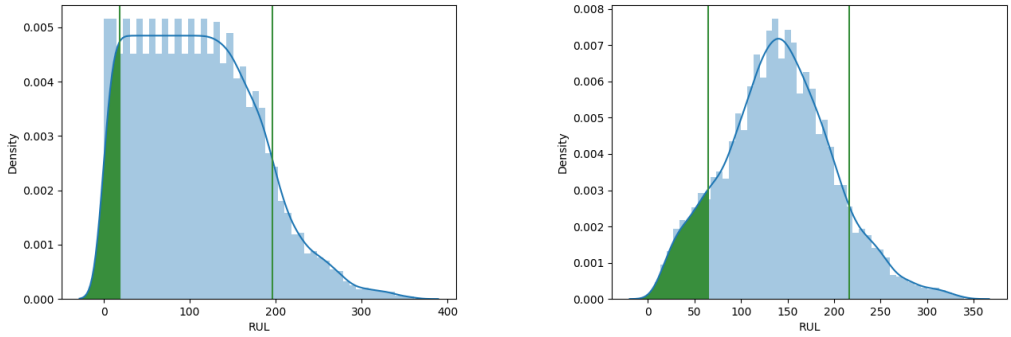
Experiments

To demonstrate the effectiveness of our solution, we conducted a series of experiments in two application domains: predictive maintenance and botnet attack detection. In each experiment, we considered both honest clients with limited datasets and malicious nodes with intentions to disrupt the training process or introduce backdoors into the global model. We compared TruFLaaS against two benchmarks: a conventional baseline with no validation mechanisms and TrustFed [94], a framework designed for fair and trustworthy federated learning.

To ensure fairness in the predictive maintenance use case, we used the same FL model configuration. The model consists of an input layer with `input_size` x 24 input neurons for each window, followed by two middle-dense layers with 24 × 24 neurons each. The output dense layer has 24 × 1 neurons. ReLU activation functions are applied to all layers, and weight adjustments are made using the stochastic gradient descent (SGD) optimizer. We evaluated the model's accuracy using Mean Absolute Percentage Error (MAPE) after the final aggregation. Additionally, we calculated the Mean Absolute Error (MAE) to validate partial models and identify the most beneficial ones for training.

Table 5.3: Botnet Attack Detection - TruFLaas experiments results.

Exp.	Nodes %	Cross Entropy	Accuracy %	Precision %	Recall %	F1 %
#1	0	0.549	77.51	79.31	75.77	73.92
	10	0.461	75.83	74.15	76.42	70.51
	25	0.419	77.5	75.86	79.11	72.20
#2	0	0.390	77.55	81.13	79.47	73.38
	10	0.409	75.83	78.12	76.81	70.7
	25	0.387	74.16	71.43	77.53	69.39
#3	0	0.41	77.39	70.31	77.39	71.95
	10	0.41	77.38	70.29	77.38	71.94
	25	0.42	76.43	82.04	76.43	70.74



(a) NASA Turbofan Jan Engine - 10^{th} percentile distribution of training data.

(b) NASA Turbofan Jan Engine - 10^{th} percentile distribution of testing data.

Figure 5.7: Distribution of 10^{th} percentile in training and testing data.

However, TrustFed had limited validation against multiple datasets and models. Therefore, for the botnet attack detection scenario, we developed a novel FL model consisting of four layers. The model's architecture includes two Dense layers with ReLU activation functions, comprising 64 and 32 neurons, followed by a Dropout layer with a 0.2 rate to prevent overfitting. The final layer is a Dense layer with 11 neurons, one for each class to be predicted, utilizing a softmax activation function. Since this use case involves multi-class classification, we employed categorical cross-entropy as the loss function. In Table 5.3, we present the final evaluation metrics for the conducted experiments.

1) *Heterogenous Data Distribution*: First, we consider a scenario where the distribution of

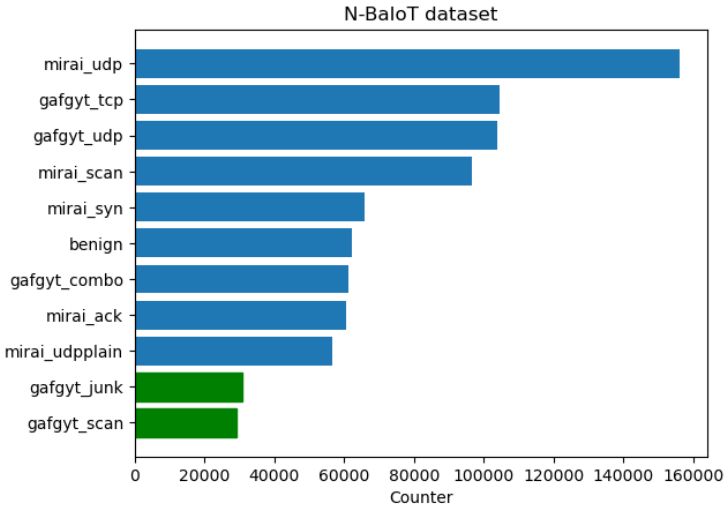


Figure 5.8: N-BaIoT - Types of botnet attacks and their occurrences.

data, among honest clients, is heterogeneous. Hence, some nodes have more or fewer data samples than others. Having clients with heterogeneous data distribution is one of the major cases that justifies the adoption of FL: this situation is quite common in industrial environments since the size of enterprises directly affects the amount of data generated. We run this type of experiment while varying the number of participants and with different percentages of nodes having augmented data.

2) *Heterogeneous Data Distribution on Rare Cases*: This experiment is a specific variation of the previous set. In the context of FLaaS, clients may have a particular interest in a specific subtask, such as predicting Remaining Useful Life (RUL) below a certain threshold or identifying a specific type of botnet attack. We are specifically focusing on a deployment scenario where some nodes lack data related to particular classes of events, which we refer to as "rare cases." For instance, there might be smart manufacturing enterprises that have not encountered the breakdown of certain machinery or have never been subjected to a specific type of attack.

In this experiment, for the predictive maintenance use case, we differentiate the data records based on their RUL values. More precisely, we categorize samples with low RUL values into the *rare records* category by isolating them from the rest of the data. We identify this subset of data through statistical analysis, calculating the 10th percentile values

for both the training and validation sets. Since NASA data do not adhere to a normal distribution, we perform a standardization process to enable us to use the z-score table for precise calculations of areas for any given normally distributed populations. Mathematically, the standardization operation is described by the formula:

$$z = \frac{x - \mu}{\sigma} \tag{5.4}$$

Here, z represents the $z - scorevalue$, x is the observation value, μ is the mean of the distribution, and σ is the standard deviation of the distribution. Figures 5.7a and 5.7b provide visual representations of the obtained percentiles on the training and validation datasets, respectively.

For the botnet attack detection use case, we employ a multiclass classification approach. Consequently, we identify the two attack classes with the lowest occurrence as "rare cases." Specifically, as indicated in Figure 5.8, these classes correspond to junk and scanning attacks.

Our experiments involve variations in the number of nodes that lack data on rare cases and the strategies for discarding nodes. We test four different strategies using two validation sets: one containing only rare cases (Rares) and another with a data distribution identical to the global test set (Overall). The first strategy involves discarding nodes that perform poorly on the validation set containing only rare cases. The second strategy entails discarding nodes that perform poorly on the second validation set. The third strategy requires the removal of nodes that perform poorly on both the first and second validation sets. Finally, the fourth strategy involves discarding nodes that perform poorly on either the first or second validation set.

3) *Model Forging Attack*: Safeguarding the security and reliability of FL platforms is of crucial concern, given their extensive utilization across diverse domains. Model forging attacks pose a significant threat, where a malicious participant may deliberately construct a partial model with the intent to insert backdoors into the global model or disrupt the training process altogether. To evaluate the robustness of TruFLassS against such attacks, we carried out a series of experiments involving varying percentages of malicious nodes. In these experiments, akin to the approach taken in TrustFed, we simulated the behav-

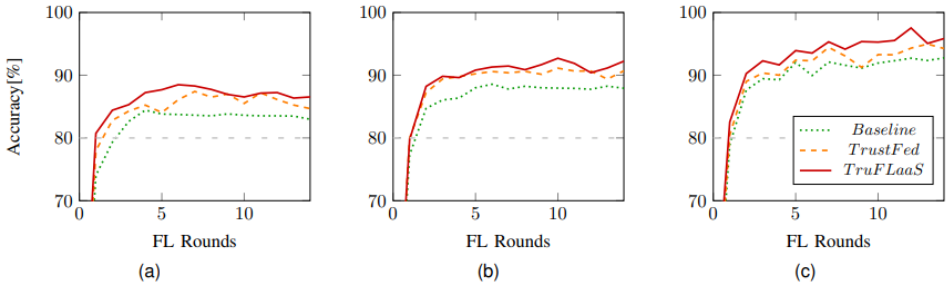


Figure 5.9: Predictive Maintenance: Heterogeneous Data Distribution - Accuracy comparison of different node selection strategies with 0 nodes having augmented data (a), 10 nodes having augmented data (b), and 25 nodes having augmented data (c).

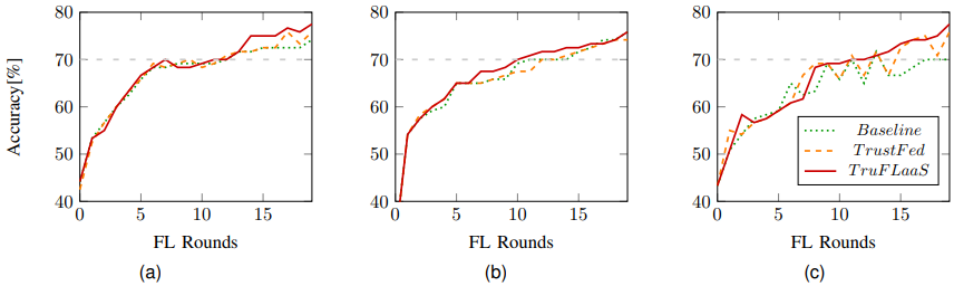


Figure 5.10: Botnet Attack Detection: Heterogeneous Data Distribution - Accuracy comparison of different node selection strategies with 0 nodes having augmented data (a), 10 nodes having augmented data (b), and 25 nodes having augmented data (c).

ior of malicious nodes by conducting training on data deliberately infused with random noise, thus compromising the model's integrity.

5.3.4 Results

The experimental results reported in this section show that our solution outperforms conventional baselines and TrustFed under all the considered circumstances.

1) *Heterogenous Data Distribution*: TrustFed exclusively consolidates partial models that exhibit accuracy within the confines of the interval determined by the vicinity of the mean and standard deviation. Nonetheless, the TrustFed methodology tends to overlook clients whose data distribution is heterogeneous, enabling them to generate high-quality partial models that surpass the performance constraints of this interval. Conversely, Tru-

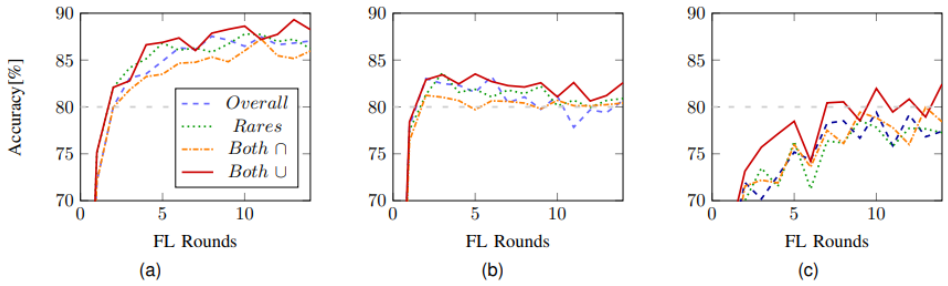


Figure 5.11: Predictive Maintenance: Heterogeneous Data Distribution on Rare Cases - Accuracy comparison of different node selection strategies with 0 nodes without rare data (a), 10 nodes without rare data (b), and 25 nodes without rare data (c).

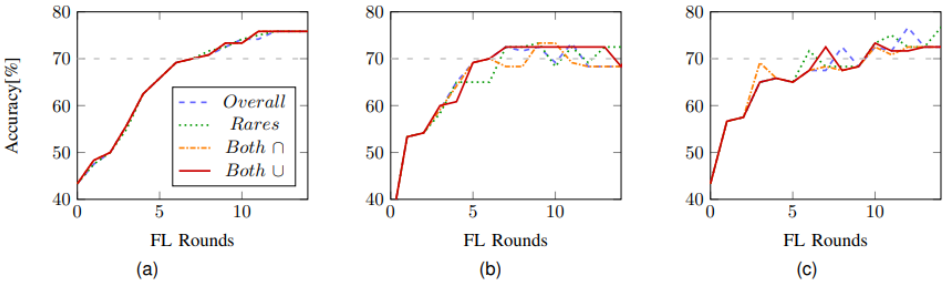


Figure 5.12: Botnet Attack Detection: Heterogeneous Data Distribution on Rare Cases - Accuracy comparison of different node selection strategies with 0 nodes without rare data (a), 10 nodes without rare data (b), and 25 nodes without rare data (c).

FLaaS adopts a different approach, discarding partial models whose performance falls below a predefined threshold. During the aggregation phase, TruFLaaS incorporates all partial models boasting accuracy greater than the lower boundary within the range defined by the mean and standard deviation. The graphical representations in Fig. 5.9 and 5.10 clearly illustrate how TruFLaaS outperforms TrustFed by effectively recognizing clients' contributions, especially those with significantly larger datasets.

TruFLaaS not only attains the desired accuracy more rapidly than other approaches but also exhibits a growing advantage as the number of nodes with augmented data increases.

2) *Heterogeneous Data Distribution on Rare Cases*: In this particular experiment, we refrained from comparing TruFLaaS with TrustFed, as the latter does not differentiate be-

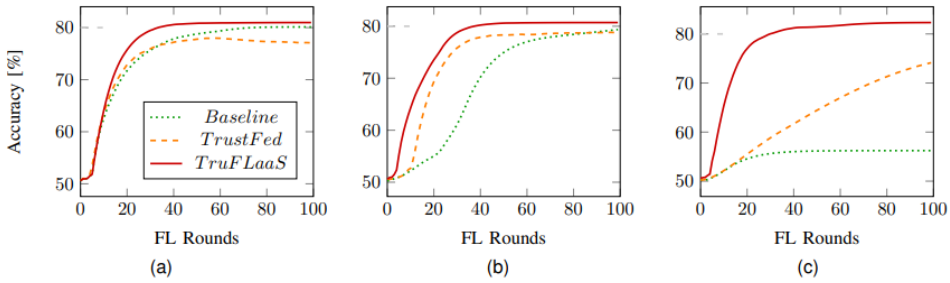


Figure 5.13: Predictive Maintenance: Model Forging Attack - Accuracy comparison of the compared approaches with 0 malicious nodes (a), 10 malicious nodes (b), and 25 malicious nodes (c).

tween rare and common cases, and our approach would unquestionably demonstrate superior performance.

The results of this experiment, illustrated in Fig. 5.11 and 5.12, vary based on the number of nodes lacking rare data and the strategy employed to exclude nodes with inadequate performance. In the predictive maintenance scenario, the first two strategies, namely, removing nodes with subpar performance on the validation set containing only rare cases and eliminating nodes with unsatisfactory performance on the validation set mirroring the test set's distribution, outperform the other aggregation strategies under consideration.

In the context of botnet attack detection, our solution exhibits resilience even as the number of nodes without rare data increases. Notably, the growing presence of heterogeneous nodes does not adversely impact accuracy.

3) *Model Forging Attack*: In the predictive maintenance use case, given that TrustFed's FL configuration did not yield satisfactory results in comparison to our solution, we extended the number of FL rounds to 100. The clear distinction in performance between TruFLaaS and TrustFed when exposed to model forging attacks, while varying the number of malicious nodes, is vividly depicted in Fig. 5.13 and 5.14.

This difference in performance primarily stems from TrustFed's use of the mean and standard deviation for outlier detection, making it less reliable in the presence of extremely large or small values. For instance, an outlier with exceptionally poor performance could significantly skew the overall mean. In such cases, TrustFed may end up accepting out-

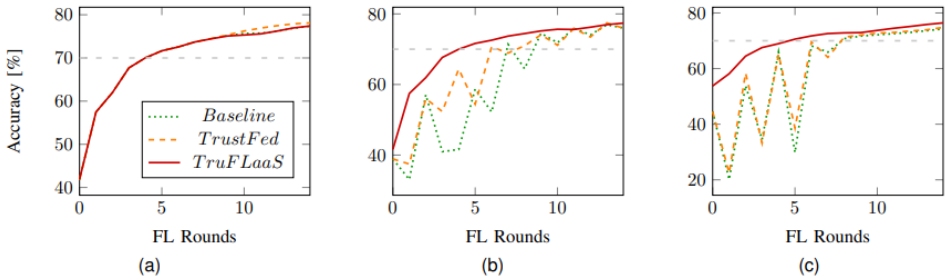


Figure 5.14: Botnet Attack Detection: Model Forging Attack - Accuracy comparison of the compared approaches with 0 malicious nodes (a), 10 malicious nodes (b), and 25 malicious nodes (c).

liers with performance that should be considered unacceptable under normal conditions. Additionally, TrustFed's failure to weigh partial models means that aggregating an outlier can entirely disrupt the progress achieved in training up to that point. Notably, in the predictive maintenance experiment, TrustFed even performs worse than the baseline with zero malicious nodes. This could be attributed to TrustFed's practice of removing nodes that attain performance levels significantly above the average.

Conversely, TruFLaaS employs a more robust outlier detection algorithm, consistently achieving similar performance across all scenarios. Furthermore, the use of weights based on the number of accepted transactions (i.e., the level of trust) ensures that sporadic errors during the validation process have minimal impact on the global model, preventing disruptions to the entire training process. These findings offer valuable insights into the robustness of our proposal and demonstrate its effectiveness in safeguarding against model forging attacks.

6 Authorization Revocation

A verifiable credential is a novel standard that securely proves a statement about an entity, without relying on centralized authorities. Thus, they are considered a key technology to establish mutual trust among entities in highly distributed environments such as Cloud-to-Thing Continuum. However, since statements can change over time, i.e. a device using a chip found with a vulnerability, how to efficiently revoke them is a major concern. In this chapter, we mainly focus on the revocation of VCs in IoT networks. Specifically, we introduce a novel approach that leverages an ECC-based accumulator. We evaluate our revocation scheme in several configurations, showcasing its effectiveness in efficiently revoking VCs while demanding minimum computing and storing capabilities.

6.1 Trust in IoT Networks

Recently, we have been witnessing the outstanding widespread of IoT devices [9], which are expected to surpass 125 billion by 2030 [57]. In such settings, the lack of *trust* [73] is one of the major concerns that limit the full usage of data and collaboration among different entities. Third-party organizations do not rely on external data for decision-making and devices do not trust each other. Therefore, increasing trust in IoT networks would remarkably contribute to leveraging a larger amount of data, leading to the development of next-generation services [100].

At the state-of-the-art, PKI is the most widespread technology adopted to identify devices and establish mutual trust [52, 53]. In this, digital identities are traditionally issued and managed by centralized CAs. However, centralized approaches present several concerns and limitations, including scalability, lack of control over own information, single point of failure, and limited interoperability.

To address these concerns, decentralized approaches are increasingly attracting the interest of the industrial and academic worlds. In this direction, the W3C has formalized

and standardized DIDs and VCs as viable solutions for promoting decentralization. A DID is a unique identifier that resolves to a DID document containing public keys, enabling the proof of ownership over specific data. On the other hand, a VC is a statement about an entity that can be cryptographically verified by a third party. VCs offer an alternative to traditional digital certificates for establishing trust and verifying the authenticity of attributes. The interest in VCs has experienced significant growth in recent years. For instance, VCs are considered a key technology by the European regulation for "electronic IDentification, Authentication, and trust Services" (eIDAS) to ensure secure and remote identity proofing [88]. DIDs and VCs have the potential to efficiently implement the concept of Identity of Things (IDoT) [58] introduced by the Kantara Initiative. Each device is assigned a DID that enables proving certain capabilities, through VCs, to other devices.

As for digital certificates, revocation of VCs for malfunctioning or compromised devices is a critical operation for the trust of the whole network. A notable example is the chip vulnerability on RSA key generation, as highlighted in CCS'17 by Namec et al. [84], which resulted in the revocation of more than 700K certificates for devices using that chip. In IoT networks, the presence of heterogeneous devices, some with limited computing and storage capabilities, adds complexity to the revocation process. Additionally, IoT networking can be constrained as well and characterized by limited bandwidth, low transmission range, dynamic topology, and unreliable connectivity. These factors pose challenges to the adoption of traditional revocation mechanisms like the OCSP [96] and CRLs [32], which respectively assume reliable connections and large amounts of memory. Despite the growing interest in DID and VCs [48, 70], the development of efficient approaches for VC revocation remains in its early stages, as evidenced by the lack of a dedicated W3C standard [93].

6.2 Efficient Revocation of Verifiable Credentials in IoT Networks

To overcome revocation challenges in IoT networks, we introduce EVOKE [71], a novel solution designed to address the efficient revocation of VCs in IoT networks. Our approach allows devices to establish trust while minimizing computing and storing overhead. To achieve this, EVOKE leverages an ECC)-based accumulator [115], which ef-

Table 6.1: EVOKE Notation.

Symbol	Description
$\mathcal{N}$	Set of IoT Networks
$\mathcal{I}$	Set of issuers
$\mathcal{ACC}$	Set of accumulators
j	Geographical area
i_j	Single issuer
D_j	Set of IoT devices in j
d_i	Single IoT device
W_j	Set of witnesses in j
w_i	Witness of device d_i
b	DLT

ficiently aggregates multiple data into a unique element, and its size remains constant regardless of the number of embedded elements. Additionally, proving membership in the accumulator demands constant verification time. These characteristics ensure that EVOKE maintains efficiency even when the number of credentials and devices within the network grows.

6.2.1 System and Threat Model

In this paper, we consider IoT networks $\mathcal{N}$ that comprise issuers $\mathcal{I}$, accumulators $\mathcal{ACC}$, a distributed ledger b, and IoT devices $\mathcal{D}$. In real-world scenarios, devices interact within specific geographical areas. Thus, without loss of generality, we assume that a geographical location j represents an IoT network $n_j \in \mathcal{N}$. Each IoT device $d_i \in D_j \subset \mathcal{D}$ in n_j has a VC issued by the issuer $i_j \in \mathcal{I}$. To manage VCs, i_j accumulates all the valid VCs of j, i_j accumulates them in $a_j \in \mathcal{ACC}$. Table 6.1 summarizes the notation used in the EVOKE system model and security analysis. In the design of EVOKE, we make the following assumptions:

1. The issuer i_j possesses sufficient computing capabilities, including an adequate level of processing power, memory, and necessary software/tools, to successfully build the accumulator a_j, remove VCs, and generate witnesses W_j for all $d_i \in D_j$;

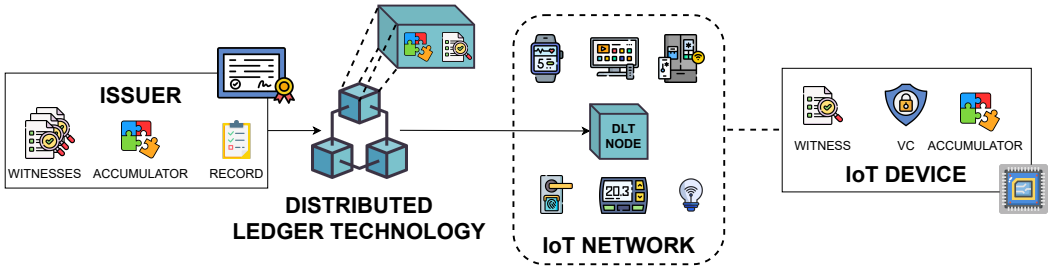


Figure 6.1: EVOKE architecture. The dotted lines represent a closer examination of revocation information shared on the DLT and data maintained by each IoT device.

2. each device d_i is initially provided with a valid VC, a_j, w_i, its cryptographic material and the public key of i_j, which has issued the VC. Through the public key of i_j, d_k can establish secure and authenticated communications.

Components

In this section, we provide a detailed explanation of EVOKE's components, whose architecture is depicted in Figure 6.1.

- *Issuer*: the issuer i_j is responsible for issuing and revoking VCs. It records all issued VCs and IoT devices within the network. The issuer operates independently from the rest of the network and only communicates through the DLT. We assume it possesses sufficient computing capabilities for managing VCs, including the generation and update of the accumulator a_j and witnesses W_j.

- *DLT*: b serves as a secure, distributed data source for exchanging information (i.e., a_j and W_j), between the i_j and D_j. Each n_j has at least one DLT node equipped with adequate resources to maintain a consistent and up-to-date version of b. In our work, we utilized a DAG-based ledger, DAG for brevity. The adoption of a DAG offers transparency and immutability of shared data while exhibiting enhanced performance in terms of latency and energy consumption that make it the best choice for IoT environments [42].

- *IoT Device*: $d_i \in D_j$ vary in computational power, resource consumption, storage capacity, and networking capabilities. Each IoT device d_i is equipped with suf-

ficient storage for essential components such as the a_j, d_i, a VC, DID, and associated information. Additionally, each d_i possesses the minimum cryptographic functions necessary to establish trust within the IoT network n_j.

Threat Model

The security of our scheme relies heavily on the secure implementation of the proposed system. We assume that the a_j has been constructed securely, ensuring its collision resistance. To assess the security of our system, we employ the Dolev-Yao model adversary [37], who possesses the capability to eavesdrop on, intercept, or inject an arbitrary number of messages. Furthermore, we also consider that the attacker can compromise the entities composing n_j. The primary objective of the adversary is to acquire a valid VC, thereby gaining the ability to establish trusted communications. This goal can be accomplished by compromising i_j, D_j, or communication links. We identify the following threats:

- **Compromised Issuers**: Gaining control over the issuer i_j gives the adversary the capability to arbitrarily issue valid VCs as well as revoke valid ones. The attacker can generate VCs with false or misleading claims, improperly granting permission to entities within the system. Furthermore, the adversary may invalidate legitimately issued VCs, resulting in a denial of services (DoS) or disruption of the established trust relationship.

- **Compromised IoT Devices**: IoT devices D_j act as the entry point to IoT networks and can be vulnerable to a range of attacks if compromised by adversaries. The attacker may identify vulnerabilities in the firmware of IoT devices to gain unauthorized access or control over other devices. Devices can be compromised in a variety of ways, such as malware distribution through firmware updates or the devices may have already been manufactured with compromised hardware or software components in case the adversary has direct access.

- **Compromised Communication**: In n_j, D_j are not obligated to implement encryption mechanisms, making it relatively easy for the adversary to eavesdrop on the traffic. For example, the attacker may use the intercepted VC to spoof the identity of the device and establish trust on its behalf.

6.2.2 EVOKE Protocol

The proposed approach allows IoT devices D_j to securely communicate with minimal overhead. Specifically, each device d_i maintains the accumulator a_j, which comprises all the valid VCs, and a witness w_i that demonstrates the validity of its credential. The value of a_j and the witnesses W_j are computed by the issuer i_j and initially provided to all $d_i \in D_j$. However, when a VC is revoked both a_j and W_j have to be updated. Using just two values, which require only a few bytes, remarkably reduces the revocation management overhead both in terms of storage and distribution. The remainder of this section provides the full details of our method.

Preliminaries

The accumulator a_j is the key element of our proposal that enables efficient management of VCs. In this subsection, we provide all the details related to its usage in EVOKE.

Proof of membership: When using a revocation list, two devices d_i and d_k that have to authenticate each other present their VCs and verify whether the received one is included in the list. Similarly, to use a_j to revoke VCs, d_i needs to present proof of *membership* named witness. In this way, the receiver applies a verification function that allows it to state whether the VC has been included in a_j.

Inputs of the accumulator: Values in a_j are derived from the hash conversion of the valid VCs. Therefore, the security of a_j relies heavily on the chosen hash function. In our case, we use the widely adopted SHA-256 hash function, which is considered to be secure and collision-resistant.

Functions: The following functions are used to implement the proposed approach:

- $a_{info} \leftarrow \mathsf{Setup}(\mathsf{K})$: setups the parameters of $\mathsf{a_j}$ by taking a set of public values K, which represent the random generators of the elliptic curve.

- $a_j \leftarrow \mathsf{ComputeAccumulator}(\mathsf{a_{info}}, \mathsf{V})$: accumulates a set of valid verifiable credentials V by mapping them onto the curve. This function is also executed to update a_j when V is modified, as VCs can be revoked and newly valid ones can be issued.

- $a_j^t \leftarrow \mathsf{RevomeFromAccumulator}(\mathsf{a_{info}}, \mathsf{a_j^{t-1}}, \mathsf{V})$: removes a set of invalid verifiable credentials V. This function is also executed to update a_j.
- $w \leftarrow \mathsf{ComputeWitness}(\mathsf{a_j}, \mathsf{vc_i})$: generates a witness value $\mathsf{w_i}$ for a valid verifiable credential $\mathsf{vc_i}$, providing proof of its membership within $\mathsf{a_j}$.
- $0, 1 \leftarrow \mathsf{Verify}(\mathsf{a_j}, \mathsf{vc_i}, \mathsf{w_i}, \mathsf{pk_i})$: the verifier executes this function to authenticate $\mathsf{d_i}$. It verifies whether $\mathsf{w_i}$ provided by $\mathsf{d_i}$ is accumulated in $\mathsf{a_j}$ and uses the device's public key $\mathsf{pk_i}$ to ensure that $\mathsf{vc_i}$ is associated with $\mathsf{d_i}$.
- $a_j^t \leftarrow \mathsf{UpdateAccumulator}(\mathsf{a_{info}}, \mathsf{a_j^{t-1}}, \mathsf{V})$: removes a set of invalid verifiable credentials V through the $\mathsf{RevomeFromAccumulator}()$ function. In case, a new set of valid verifiable credentials V is released, they are included in a_j using the $\mathsf{ComputeAccumulator}()$ function.
- $w^t \leftarrow \mathsf{UpdateWitness}(\mathsf{a_j^t}, \mathsf{vc_i})$: updates the witness corresponding to a valid verifiable credential $\mathsf{vc_i}$. When $\mathsf{a_j}$ is updated, previously disseminated $\mathsf{W_j^{t-1}}$ become invalid. This function takes the updated value $\mathsf{a_j^t}$, a verifiable credential $\mathsf{vc_i}$, and returns a valid witness $\mathsf{w_i^t}$ corresponding to $\mathsf{vc_i}$.

Protocol

This subsection describes the main phases of our protocol, which leverages the accumulator presented in [115]. Given two IoT devices d_i and d_k, they can validate each other as long as the accumulator a_j and their respective witnesses, w_i and w_k, are updated. To securely share a_j among all $d_i \in D_j$, i_j includes the accumulator a_j, as well as the witnesses W_j, in self-signed VCs and publishes them on b. This approach enables D_j to verify the authenticity before updating.Furthermore, if d_i is outdated and cannot directly download the freshly issued a_j and w_i, the updated d_k can share the refreshed VCs needed to establish mutual trust. The proposed protocol supports mass and offline revocation, which are defined as follows:

Definition 3 (Mass Revocation)**.** *Mass revocation is the capability of efficiently revoking a large number of VCs without impacting the issuer and device overhead. A protocol designed for mass revocation ensures that the memory size of the underlying data structure as well as the verification time remain unaffected by the number of revoked VCs.*

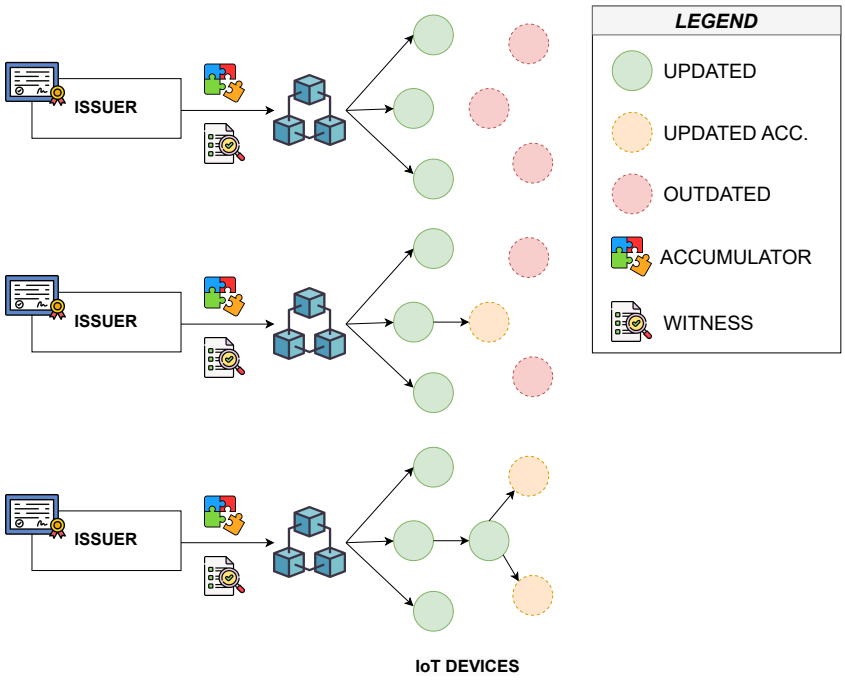


Figure 6.2: Updating accumulator and witnesses.

Definition 4 (Offline Revocation)**.** *Offline revocation refers to the capability of devices to receive and apply updates even when they have been disconnected from the network, without burden on network resources. A protocol that supports offline revocation is designed to minimize the amount of data transmission overhead required to facilitate these updates.*

Setup. In the setup phase, i_j securely generates the accumulator's parameter K and initializes a_j by executing the $\mathsf{Setup}()$ function. Valid VCs are accumulated into a_j through the $\mathsf{ComputeAccumulator}()$ function. For each valid VC, i_j generates a hash value, which serves as a unique identifier, which is then used to obtain the corresponding representation on the curve. After computing the final a_j value, i_j generates the witnesses $w_i \in W_j$ for each valid VC by using the $\mathsf{ComputeWitness}()$ function. W_j provides cryptographic proof of the inclusion of the VC in a_j. Initially, each d_i receives a_j, a valid VC, and its corresponding w_i. These components are essential for subsequent operations.

Authentication. During the authentication, d_i imitates the communication with d_k by sending the timestamp of a_j. The issuer associates a timestamp with the accumulator that serves as a means to synchronize the devices and ensure consistency. Device d_k compares

the received timestamp with that of its own accumulator. Matching timestamps indicate that both d_i and d_k are operating with the same version of a_j, and they can proceed with the authentication. However, if the timestamps differ, it implies that one of the devices has an outdated a_j. In this case, the device with the older timestamp updates its a_j to the latest version. Once synchronized, d_i and d_k exchange VCs and the corresponding witnesses, namely w_i and w_k. Both devices hash the received VC and utilize the Verify() function to check if the hashed VC provided by the other device is included in a_j. This verification step ensures that the VCs are valid and have not been tampered with. If both VCs are found within a_j, it indicates their membership and authenticity. Consequently, the devices can trust each other and establish a secure and trusted communication channel.

Update. Over time, the issuer i_j may need to revoke certain VCs due to device compromise or faults. Revoking VCs entails updating the accumulator a_j and the corresponding witnesses W_j for the IoT devices that are still functioning correctly. Figure 6.2 offers a high-level overview of our protocol while updating an IoT network. To revoke VCs, i_j executes the UpdateAccumulator() function that removes invalid credentials from a_j.Once the updated a_j is obtained, i_j uses the novel version to generate a new W_j for the remaining valid VC. The UpdateWitness() function is employed to calculate the updated witnesses based on the modified a_j. These witnesses provide cryptographic proof of the membership of the VCs within a_j. Subsequently, i_j embeds the updated information, including the newly generated W_j, within VCs signed by i_j to ensure their authenticity. These modified VCs, containing the refreshed witnesses, are then shared through a DAG b. By distributing the updated VCs with their corresponding witnesses, other devices in the network can be informed about the revoked credentials and have access to the latest valid set. This mechanism ensures that all devices have an up-to-date view of the trusted credentials within the system, promoting secure and reliable communication.

In an IoT network n_j, offline updates are essential for devices with intermittent connectivity or power-saving hibernation. EVOKE leverages VCs to share a_j and W, enabling devices to update even when not directly connected. When an updated device d_i encounters an outdated device d_k, it can facilitate the update process by sharing a VC that contains a more recent version of a_j. Depending on the networking capability of d_k, there are two possible approaches: *direct retrieval* and *indirect retrieval*. If d_k has suffi-

Algorithm 2 `HandleUpdates`. Function executed to update $d_i \in D_j$. If d_i becomes outdated, it can still receive revocation information from updated devices.

Input: D_j, b
Output: updated D_j
for each $d_i \in D_j$ **do**
    d_i interacts with some $d_k \in D_j$ with $i \neq k$
    if a_j^i is older than a_j^k **then**
        $a_j^i \leftarrow a_j^k$
        if d_i has sufficient connectivity **then**
            $w_i \leftarrow \text{DirectRetrieval}(b, d_i)$
        else
            if d_k has sufficient connectivity **then**
                $w_i \leftarrow \text{IndirectRetrieval}(b, d_k, d_i)$
            end if
        end if
    else
        if a_j^k is older than a_j^i **then**
            $a_j^k \leftarrow a_j^i$
            if d_k has sufficient connectivity **then**
                $w_k \leftarrow \text{DirectRetrieval}(b, d_k)$
            else
                if d_i has sufficient connectivity **then**
                    $w_k \leftarrow \text{IndirectRetrieval}(b, d_i, d_k)$
                end if
            end if
        end if
    end if
end for

cient networking capability, it directly retrieves the corresponding w_k from b. Otherwise, it obtains w_k from d_i or another device with satisfactory network connectivity. This indirect retrieval process ensures that d_k receives the necessary w_k to accompany the updated a_j. Importantly, both a_j and w_k should be included within a VC to facilitate the sharing process. By utilizing VCs to share a_j and W, the EVOKE system ensures that updates are always trusted, even if not directly provided by i_j. Algorithm 2 shows how updates are handled in n_j.

Discussion. EVOKE's ability to handle intermittent connectivity and hibernation ensures that IoT devices D_j can remain up-to-date, thereby maintaining the system's in-

tegrity and security. However, when two outdated devices, d_i and d_k, interact, it is important to consider two possible scenarios that require further attention.

- *Consistent accumulator version*: If both devices d_i and d_k have the same version of the accumulator a_j, they are not able to detect any missed updates. Consequently, the devices trust each other and proceed with mutual authentication. Any necessary updates are applied when these devices encounter an updated device in n_j.

- *Inconsistent accumulator version*: When both the devices are outdated, but possess different versions of a_j, additional steps are required. Let's assume that d_k has the older version of a_j. If d_k is capable of implementing direct retrieval, it retrieves w_k and recognizes that it corresponds to a fresher version of a_j than what was exchanged. As a result, d_k obtains the associated novel version of a_j and updates itself accordingly. On the other hand, if indirect retrieval is necessary, d_i collects w_k and performs a similar verification process. Eventually, d_i acquires the latest version of a_j and the corresponding witnesses and updates d_k accordingly. By following this procedure, devices ensure that their accumulators and witnesses are synchronized, allowing for consistent and trusted communication.

These scenarios highlight EVOKE's adaptability, allowing D_j to reconcile differences in accumulator versions and update each other for consistency. Through these mechanisms, EVOKE promotes seamless communication and guarantees the overall security and reliability of the IoT network n_j.

6.2.3 Security Analysis

In this section, we consider the threats introduced in Section 6.2.1 and analyze how EVOKE provides strong security guarantees against them.

Security of Issuer

The issuer i_j is the primary target of the adversary as it manages the accumulator a_j, which is the key data structure for handling VCs. We examine three potential attack scenarios and corresponding countermeasures:

1. *Securing the Issuer's Environment*: i_j is deployed in a secure environment that imposes restrictions on direct external communication. Valid VCs are accumulated within the issuer's secure perimeter, and a_j is shared with n_j through a DAG b. This setup makes it significantly challenging for the attacker to gain unauthorized access a_j from external sources.

2. *Compromise of Accumulator Security Measures*: Despite robust defense mechanisms in place, the adversary may still attempt to compromise the security measures protecting a_j and acquire the setting parameters $\mathsf{a}_{\mathsf{info}}$. However, even if the adversary gains access to these parameters, it does not provide them with an advantage in attacking the revocation management process. To undermine our approach, attackers would need to obtain the key pair associated with the issuer DID and use it to sign fraudulent VCs that comprise the accumulator a_j and witnesses W_j.

3. *Key Pair Compromise*: If an adversary successfully obtains the issuer's key pair, immediate action must be taken to mitigate the risk. This includes migrating i_j to a new server and providing it with a fresh DID and key pair. By replacing the compromised key pair, the system maintains the integrity and security of VCs, preventing unauthorized access and fraudulent activities.

Security of IoT Devices

The adversary can compromise an IoT device $d_i \in D_j$, enabling them to launch various attacks within the IoT network n_j. d_i provided with a valid VC and w_i can establish trusted communications with other entities that place trust in i_j. Therefore, i_j is responsible for identifying and detecting malicious activities, ensuring that VCs associated with compromised devices are removed from a_j. Several approaches have been proposed to detect compromised devices and malicious activities [27, 69, 124], however, these challenges lie beyond the scope of this paper. i_j updates W_j and a_j, distributing them to D_j. As a result, malicious $d_i \in D_j$ will not be provided with valid witnesses, thereby preventing the attacker from successfully bypassing the revocation mechanism using a compromised d_i or its stolen VC.

Security of Communications

In n_j, the traffic may not be encrypted, thus, further broadening the attack surface. We consider two possible attack scenarios and corresponding countermeasures occurring during the update phase:

1. *Selective VC Fowarding*: One attack scenario involves the attacker intentionally not forwarding the VC containing the updated version of the accumulator a_j to outdated devices. As a result, outdated devices may continue to trust devices whose VCs are no longer valid. However, it is important to note that an outdated device can still be updated by an honest device during communication initiation. Additionally, to address the concern of the adversary providing an outdated version of a_j, devices only update the value of a_j if the associated timestamp is fresher than the previous one. By incorporating timestamp-based checks, devices can ensure that updates are only applied when newer information is available, mitigating the risk of relying on outdated VCs. Furthermore, an attacker may not provide the latest version of a_j; instead, they may provide a version that includes their revoked VC but is fresher than that of an outdated device. In such cases, if the device supports direct retrieval, it can directly detect the deceptive attack when collecting its updated witness. Otherwise, the adversary provides the device with the witness corresponding to that version of a_j, and a mutual trust is established. However, it is worth noting this is an extreme scenario and it is limited over time since, as shown in Section 6.2.4, most of the devices are updated within the first hour. Once the devices receive the updated a_j, the attacker can no longer deceive them since their VC will no longer be included in the accumulator.

2. *VC Theft and Masquerading*: The attacker can steal the VC comprising the updated w_i of d_i, attempting to masquerade as the legitimate device. However, since the adversary does not possess the key pair associated with d_i, they cannot prove the ownership of the VC and establish trusted communication. Furthermore, in case of transmissions that involve sensitive information, data can be encrypted using the public key of the receiving device.

6.2.4 Evaluation

In this section, we present a comprehensive evaluation of our approach. We designed a series of experiments to thoroughly assess the effectiveness and applicability of EVOKE. In particular, we considered the following scenarios:

- **Commodity IoT Devices**: to gauge the practicality of implementing EVOKE in real-world scenarios, we conducted experiments on various off-the-shelf IoT devices.
- **Hybrid Network Settings**: since not all commodity devices support running customized code directly, we performed a set of experiments on an IoT network that comprises both off-the-shelf devices and Raspberry Pis to explore more customizable and broader scenarios.
- **Large-scale Analysis**: as it would be impractical to evaluate our proposed approach on thousands of devices, we devised a set of experiments to simulate a large-scale IoT network and evaluate the scalability of EVOKE.

The results of these experiments, along with detailed explanations, are provided in the subsequent subsections. Through this comprehensive evaluation, we aim to demonstrate the effectiveness, feasibility, scalability, and versatility of EVOKE in various IoT deployment scenarios.

Implementation Setup

We developed a prototype of EVOKE[1] to evaluate its runtime performance and test its core functionalities. We utilized JavaScript and WebAssembly (WASM) [12] for the implementation in commodity IoT devices. To ensure consistency across all experiments, the same functions were also employed in all scenarios. To enable cryptographic operations, we leveraged the `crypto-wasm-ts` library developed by docknetwork [36]. This library provides a comprehensive set of cryptographic primitives and algorithms implemented using WASM. It ensures efficient and performant execution of cryptographic tasks within EVOKE. For managing DIDs and VCs, we utilized the `identity-wasm/web`

[1] https://github.com/evokevc/EVOKE

(a) Star network topology (b) Mesh network topology

Figure 6.3: Topologies used in our hybrid networks experiments.

library from IOTA [46]. This library offers robust functionality for creating, managing, and verifying DIDs and VCs. Finally, to handle the DLT, we employed the IOTA Tangle [91], a DAG that is specifically designed to cater to the unique requirements of IoT applications [117, 118]. All the results have been averaged over 1K runs. To ensure transparency and reproducibility, we have made our prototype implementation, along with all the experiments, available on GitHub[2].

Commodity IoT Devices

In these experiments, we aim to evaluate the feasibility of using EVOKE on commodity IoT devices. For this, we set up an IoT network comprising different types of off-the-shelf IoT devices, which can be found in modern smart homes. In our study, we considered devices capable of running EVOKE's code directly. Specifically, all the devices support JavaScript and WASM. To ensure compatibility with various devices, we realized different approaches. For instance, we developed a basic application using the LG WebOS developer framework [65, 112] for the LG Smart TV. For other devices, the code was provided through a Python web server.

Table 6.2 reports the specifications of the considered devices, which feature varying resources and operating systems. Table 6.3 presents the results. In particular, we highlighted the performance of key operations. It is important to note that the size of the information is independent of the specific device used. The storage requirements for maintaining the accumulator and the witness are approximately 1.5 KB. Notably, these values, along with the execution time of the main operations, remain consistent regardless of the number of VCs involved. This demonstrates the scalability and efficiency of

[2]https://github.com/evokevc/EVOKE

Table 6.2: Commodity IoT devices used in the experiments and their specifications.

Device	Type	Processor	Memory	Operating System
LG Smart TV	Smart TV	LG Quad Core Processor	1.5GB RAM	WebOS
Amazon Echo Show 5	Home Assistant	MediaTek MT8163	1 GB RAM	Linux
Apple iPhone 12	Smartphone	Apple A14 Bionic	4GB RAM	iOS 16
Oculus Quest 2	Virtual Reality Headset	Qualcomm Snapdragon XR2	6GB RAM	Oculus OS

Table 6.3: Performance evaluation of EVOKE operations on commodity IoT devices.

Operation	LG Smart TV	Amazon Echo Show	Apple iPhone 12	Oculus Quest 2
Verify valid VC	477.44 ms	499.70 ms	12.62 ms	48.69 ms
Verify revoked VC	476.89 ms	498.67 ms	12.58 ms	47.89 ms

EVOKE across different commodity devices in terms of resource utilization and execution time.

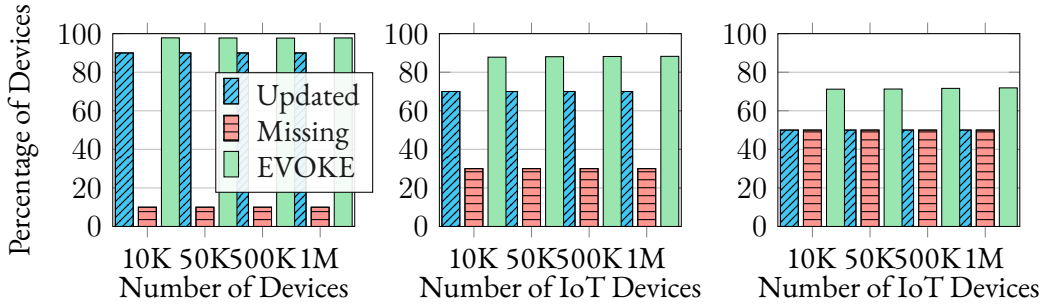
Hybrid Network Settings

In most cases, off-the-shelf devices do not provide direct access for running custom code. However, they often offer APIs that allow developers to control and interact with them. In this scenario, we considered a hybrid IoT network, which consists of both off-the-shelf devices and Raspberry Pi, serving as controllers for the devices that only provided APIs. By leveraging the capabilities of Raspberry Pi, we were able to perform EVOKE's other operations. The Raspberry Pi devices act as intermediaries, enabling the execution of EVOKE functionalities on behalf of the devices. This approach allowed us to overcome the limitations imposed by the off-the-shelf devices' restricted programmability and harness the power of EVOKE's features. Moreover, this setup ensured the seamless integration of EVOKE operations across a range of devices, expanding the potential applications and scalability of the system. In modern smart homes, advanced integration and connectivity between devices enable enhanced safety and convenience features. For

instance, consider a scenario where a smart lock is intelligently linked with a smoke detector. EVOKE allows establishing trust among the devices, enabling it to automatically unlock the door to facilitate the evacuation of occupants during a fire emergency.

In our experiments, we evaluated the performance of EVOKE in two popular network topologies for IoT protocols (i.e., Zigbee and Z-Wave) and scenarios [126]. Specifically, we considered star network topology and mesh network topology, whose deployments are depicted in Figure 6.3. These scenarios represent real-world use cases and provide valuable insights into the effectiveness and applicability of EVOKE in real IoT environments. The star topology (Figure 6.3a) consists of a central hub or controller connected to multiple peripheral devices. In a smart home scenario, it is usually represented by a home assistant (i.e., Amazon Alexa or Google Home). All communication flows through the central hub, which acts as a centralized point of control and coordination. This topology offers simplicity in management and facilitates efficient communication between devices and the central hub. However, it can pose limitations in terms of scalability and fault tolerance, as the entire network relies on the central hub. The mesh topology (Figure 6.3b) involves interconnecting multiple devices in a decentralized manner. Specifically, we considered a fully connected network, a particular case of a mesh network where each device can communicate directly with all the other devices. This topology provides robustness and redundancy, as there are multiple paths for data transmission. It can adapt well to dynamic environments and enables better fault tolerance. However, the complexity of managing and coordinating communication among numerous devices increases with the size of the network. By considering both the star and mesh topologies in our experiments, we aimed to assess the performance and suitability of EVOKE in different network configurations.

To evaluate the performance of the two network topologies, we implemented them using a local cluster consisting of five Raspberry Pi4 provided with 8GB RAM. All Raspberry Pis were connected to the same Wi-Fi network, ensuring a controlled environment for our experiments. This setup allowed us to observe the network configurations and measure latency with precision. However, due to the inherent variations in clock accuracy among different devices, it was essential to synchronize their timestamps accurately to obtain reliable metrics in our experiments. To achieve this, we utilized the Network Time Protocol (NTP) [79], a widely adopted networking protocol specifically designed for clock synchronization in distributed networks.



(a) 10% devices missing updates (b) 30% devices missing updates (c) 50% devices missing updates

Figure 6.4: Percentage of updated IoT devices by EVOKE varying the number of devices that miss updates.

Table 6.4: Performance evaluation of EVOKE operations on commodity IoT devices.

Topology	Approach	Total Latency	E2E Latency
Star Network	EVOKE	1152.7 ms	705.5 ms
	Baseline	967.7 ms	948.3 ms
Mesh Network	EVOKE	545.2 ms	307.5 ms
	Baseline	97.4 ms	91.7 ms

In Table 6.4, we present the results for the two network topologies examined in our study. We focus on the key metrics: total latency and end-to-end (E2E) latency. The total latency encompasses the overall delay in establishing trust between two updated devices. On the other hand, E2E latency specifically captures the time taken for the verification data to traverse between devices. As expected, the star network exhibits latency more than double that of the mesh network. This discrepancy arises due to the higher number of devices involved in interactions within the star network. Notably, network overhead, including synchronization delay introduced by NTP, is the main factor to affect the overall latency. To evaluate the EVOKE overhead, we compared it to a *baseline*, intended as the latencies while sending a minimum amount of data in the same network configurations without performing any operations. In the baseline, the total latency represents the delay to send a message and receive the response. On the other hand, the E2E latency is the delay in sending a minimum amount of data from one device to another. EVOKE operations for verifying VCs only contribute a few milliseconds per device, posing minimal impact on total latency. Therefore, the increased latency between EVOKE and the base-

line is due to the transfer of revocation information, however, its size is much lower than existing approaches.

Large-scale Analysis

To conduct a scalability evaluation of EVOKE, an extensive number of devices would be required. However, due to practical limitations, we implemented a large-scale network simulation of IoT devices. The simulation was run on an 11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz equipped with 4 cores and 16GB of RAM. The configurations we considered involved scaling the number of devices from 10,000 up to 1 million over a simulated week. In addition to assessing scalability, our experiments also focused on evaluating the effectiveness of offline updates. To achieve this, we categorized devices as either updated or outdated, with varying percentages of devices missing updates. Additionally, an outdated device can be normal or constrained, the former supports direct retrieval, while constrained devices can only be updated indirectly. We adopted a revocation share of approximately 0.028% of VCs per day (almost 10% yearly), based on similar studies [60] that addressed PKI-certificate revocation. This estimation was derived from the impact of the Heartbleed bug on certificates [68]. When a VC is revoked by the issuer, the accumulator and witnesses are updated and disseminated through the Tangle. However, due to various circumstances such as lack of connectivity or devices being in power-saving hibernation mode, a portion of IoT devices may not receive these updates. In our simulations, we considered different missing shares of 10%, 30%, and 50% in different experiment executions to understand the impact of outdated devices on the system. We designed interactions where each device communicates with 5 random devices within an hour. During the interactions, devices exchange their accumulators implementing the protocol described in Section 6.2. Therefore if a device has an outdated witness, the node with the older version requests an updated witness from the Tangle node or the interacting device if the capabilities are not sufficient.

In Figure 6.4, we highlight the percentage of updated IoT devices in one hour by EVOKE while varying the number of devices missing updates. `Updated` bars indicate the percentage of devices that have received the novel accumulator and witnesses, while the `Missing` bars denote the percentage of devices missing such information. As clearly shown, the percentage of updated devices is not affected by the number of devices. The

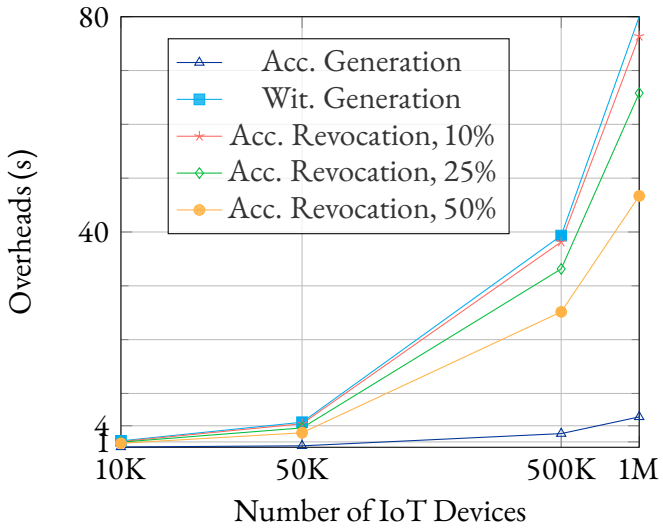


Figure 6.5: Issuer overheads for managing VCs.

experimental results demonstrate that EVOKE can update almost the whole IoT network in the first hour when the number of devices missing updates is 10%. By increasing the number of disconnected devices to 30% and 50%, the percentage of updated devices in the network reached approximately 90% and 75%, respectively.

Figure 6.5 illustrates, on a log scale, the three primary overheads encountered on the issuer side: the generation of the accumulator, the production of witnesses for the included VCs, and the updating of the accumulator. This latter comprises the removal of VCs and the generation of witnesses. The update of the accumulator was analyzed under various scenarios, considering different percentages of revoked VCs: 10%, 25%, and 50%. Evaluating EVOKE while revoking half of the issued VCs demonstrates its efficiency in handling mass revocations: the store requirements are not affected, while as the revoked VC percentage rises, the accumulator update time decreases due to fewer witnesses generated. Indeed, the graph highlights that the generation of witnesses is the most time-consuming operation. As demonstrated [18], the number of operations needed to generate w witnesses requires at least $\Omega(w)$ operations. However, even when generating 1 million witnesses, the process only requires 80K ms. It is important to note that issuers typically have more computational power, enabling a significant reduction in this overhead. Furthermore, by increasing the percentage of revoked VCs, we observe a decrease in the time required for updating the accumulator and generating the new version of the

witnesses. This result is because updating the accumulator involves the removal of VCs, which in turn reduces the number of witnesses to be generated.

Discussion

In this section, we present further considerations regarding the conducted experiments. As previously mentioned, to ensure consistency across all scenarios, including those involving commodity IoT devices, we implemented all experiments using the same JavaScript and WASM libraries. However, it is worth noting that employing diverse implementations and programming languages could potentially yield improved performance outcomes. These variations could lead to optimized execution and better resource utilization.

Commodity IoT Devices. Vendors typically restrict end users from directly running code on their devices. To demonstrate the compatibility of EVOKE with commodity devices, we developed a JavaScript/WASM application. In this approach, a device sends a request to a server, which responds with an HTML page containing our JavaScript code. This allows the execution of the primary EVOKE operations on the device itself. However, it is important to note that we had to consider commodity devices with a browser connection to run EVOKE through JavaScript/WASM. Nevertheless, we demonstrated that vendors have the opportunity to integrate EVOKE into their products.

Hybrid Network Settings. The adoption of a hybrid network, which combines both off-the-shelf devices and single-board computers such as Raspberry Pis, provides a holistic approach to exploring the potential of our solution. This setup not only demonstrates the versatility of EVOKE but also broadens its scope, making it applicable in contexts where different types of devices coexist and require efficient and secure management of VCs. Through this set of experiments, we demonstrated the applicability of EVOKE even when the code is not executed directly on the target device but rather on the controlling board. This flexibility opens up new possibilities for deploying our solution in diverse environments. Moreover, the introduction of a controller in the network further expands the potential applications of EVOKE. By incorporating a controller into the system, we can extend the utilization of VCs and, consequently, the use of EVOKE to encompass more resource-constrained devices. This capability becomes especially rel-

evant for incorporating small sensors that lack the computational power to handle VCs or any revocation mechanisms directly.

Large-Scale Analysis. This set of experiments allowed us to demonstrate two important features of EVOKE that can be achieved through the accumulator: mass and offline revocation. The size of the accumulator does not increase with the number of devices, thus, it can be employed for massively revoking VCs in any IoT network. The information stored by each device for the accumulator and witness is always in the order of 1.5 KB. Including the accumulator within a VC signed by the issuer and the reduced size of the accumulator itself enable updated devices to update the ones that missed fresh revocation information. Also when 50% of devices are missing updates, after 1 hour, around 72% of the whole network has been successfully updated. Moreover, as shown in Section 6.2.5, the network overhead of EVOKE to update information is minimal compared to other approaches.

6.2.5 Comparison to Other Approaches

This section compares EVOKE to the most widespread techniques in the revocation space and V'CER [60], which we consider the closest work in the literature. It is worth noting that these approaches are applied to PKI certificates, but they can be extended to VCs.

OCSP. In OCSP [96], the verifier sends a request to the issuer to check the validity of the presented VC. The traditional OCSP approach assumes the verifier has a fast and reliable connection to the issuer. However, this assumption may not hold in IoT networks due to the network overhead and the limited connectivity capabilities of certain devices. OCSP Stapling offers a more efficient alternative. It shifts the responsibility of querying the issuer from the verifier to the presenter of the VC. The presenter periodically queries the issuer and appends a time-stamped OCSP response, which is signed by the issuer. Thus, the verifier can obtain the necessary information about the VC's validity without additional communication. While OCSP Stapling provides a more efficient approach, it is worth noting that the stapled OCSP responses have a limited validity period and do not guarantee information freshness, which is a key advantage of this protocol. Furthermore, each OCSP request typically consumes around 4 KB of data [68], which exceeds the 1.5 KB required by EVOKE. Recently, TinyOCSP [52] managed to reduce the size

Table 6.5: Comparison of EVOKE to other approaches

Work	Storage 1M Cert/VC	Mass Revocation	Offline Revocation
EVOKE	1.5 KB	X	X
V'CER [60]	3 KB	-	X
Revoc. List 2020 [93]	125 KB	-	-
CRLite [105]	112.5 KB	-	-
Let's Revoke [63]	70 KB	-	-
TinyOCSP [52]	-	-	-

by approximately 70%; however, meeting this requirement while maintaining a reliable connection remains challenging in many IoT networks.

CRLs. When using CRL [32], devices need to store the whole list. However, in IoT networks comprising many devices and VCs, the amount of data to memorize can be extremely large. A CRL that contains only strictly necessary information for 1 million certificates would require at least a few MBs of storage [68]. Concerning VCs, the only revocation mechanism proposed by W3C is the draft namely Revocation List 2020 [93]. It associates each VC with a position in a bitstring managed by the issuer. If the binary value at a specific position is set to 1, the VC is considered revoked; otherwise, it is considered valid. With 1 million VCs, this would result in approximately 125 KB of storage. However, the use of a bitstring offers the advantage of high compressibility since a significant number of credentials often remain unrevoked, leading to long sections of repetitive bits. By employing compression techniques like ZLIB [35], the compressed size of the bitstring can be nearly halved, assuming an average compression ratio of 50% in case of mass revocation. Although a lower percentage of revoked VCs would yield a higher compression ratio, the memory storage would still exceed that of EVOKE. Indeed, EVOKE always requires that each device only stores 1.5 KB, even though the number of devices in the network may be larger. There are ongoing research efforts, such as CRLite [105] and Let's Revoke [63], aimed at reducing the storage and update overhead specifically for the Web's Public Key Infrastructure PKI. These solutions can reduce the size of the CRL to hundreds of kilobytes.

V'CER. In the literature, we find V'CER as the closest work to EVOKE. V'CER aims to tackle the challenge of efficient trust establishment on constrained networks using PKI certificates. V'CER adopts Sparse Merkle Trees (SMTs) [**10.1007/978-3-319-47560-8_13**], a type of MT that contains all possible hash values. Similar to our approach, V'CER concatenates valid certificates to create a hash root, which can be considered analogous to the final value of the accumulator in EVOKE. However, instead of witnesses, V'CER employs a small set of hashes known as the co-path, which represents all sibling nodes on the path to the root and serves as a proof of inclusion (PoI). The size of the PoI is $O(\log(n))$, where n is the number of leaves. Consequently, the storage required by devices may increase with the number of certificates. For instance, when dealing with 1 million certificates, the storage demand is approximately 3 KB, which is double the space required by EVOKE.

Traditional approaches and their enhancements are unsuitable for IoT networks due to their storage and connection requirements. Additionally, in the event of missed updates, these approaches always rely on devices directly contacting the issuer or their delegates. While V'CER presents a valuable solution tailored for constrained devices, it demands a larger amount of data per device compared to EVOKE. However, it offers the advantage of enabling devices to distributively repair their proofs in certain cases, without relying on CAs or delegates. In contrast, EVOKE is the most efficient in terms of storage requirements since it demands around 1.5KB of storage and facilitates mutual assistance among devices to update. As a result, EVOKE maintains constant storage requirements, enabling efficient mass revocation, while also empowering updated devices to bring outdated ones up to date, thus facilitating offline revocation. Other approaches may implement offline revocation, however, they employ bigger data structures that cause remarkable bandwidth overhead and make them unsuitable for constrained networks. For example, CRLs demand roughly two orders of magnitude greater than EVOKE for data transmission. An overview of the discussed approaches is provided in Table 6.5, where the storage column indicates the size required to store 1 million certificates or VCs, depending on the focus of the work.

7 Conclusions

Traditional security mechanisms are unable to adapt to the changing and evolving conditions of the Cloud-to-Thing Continuum. The lack of adaptable security mechanisms motivated the research efforts of this thesis, which has focused on threat modeling and access control. In the following paragraphs, we briefly revise the proposed security mechanisms and achieved results.

Chapter 4 presents an adaptable threat model methodology for edge computing and CPSs. Our solution leverages a Threat Catalogue, including over 150 entries, that associates with each asset a set of threats, security controls, and MTD techniques based on their capability and data involved. We showed how the proposed methodology can be applied through a case study, comprising several IoT devices that are seamlessly integrated with a gateway that manages their operations. The gateway interfaces with the cloud-based services to analyze sensory data, make informed decisions, and optimize the entire system.

In Chapter 5, we evaluate how access control policies are disseminated across multiple areas. Experimental results demonstrate that DLTS can be efficiently used to share policies while preserving their properties of immutability, confidentiality, and authenticity. Later in the same chapter, we propose a novel risk-based authorization middleware for healthcare, which is one sector that has been significantly reshaped by the Cloud-to-Thing Continuum paradigm. Access to data and resources is granted according to context information, including the health status of the patient inferred through FL. Thus, providing the needed adaptability to deal with changing contexts. Furthermore, this chapter also proposes a novel framework for enabling trustworthy FL, namely, TruFLaaS. Our solution transparently validates partial models by leveraging blockchain, smart contracts, and a DON. TruFLaaS uses smart contracts to keep the level of trust of each participant, which is used to weigh the contributions of each partial model during the aggregation phase. The extensive experimentation shows that TruFLaaS outperforms conventional

baselines and the state-of-the-art literature for the detection of malicious nodes under different relevant families of use cases, i.e. when forging an ad-hoc model to pass the validation process, or discarding low-quality models on rare data, or making a global model converge by varying the number of malicious nodes.

Finally, Chapter 6 introduces EVOKE, an efficient revocation mechanism for VCs in IoT networks. EVOKE leverages an ECC-based accumulator to revoke VCs, minimizing storage and computation overhead, which are two fundamental requirements due to the heterogeneity that usually affects the IoT environment. Since the size of the accumulator does not grow with the number of included elements, EVOKE can efficiently revoke VCs without impacting the storage consumption and the verification time. In addition, the presented protocol also enables devices to update each other in case of missing updates due to network disconnection. Experimental results demonstrate that EVOKE can be successfully deployed by commodity IoT devices (e.g., Amazon Echo Show) with a constant storage overhead of 1.5 KB. Furthermore, we also showcase its effectiveness in hybrid networks comprising IoT devices and controllers. Finally, the large-scale analysis demonstrates that even when 30% of devices missed updates, approximately 90% of devices in the entire network were updated within the first hour, proving its capability to provide offline revocation.

Acronyms

CA	Central Authority
CPS	Cyber Physical System
CRL	Certificate Revocation List
DAG	Directed Acyclic Graph
DID	Decentralized Identifier
DLT	Distributed Ledger Technology
DON	Decentralized Oracle Network
ECC	Elliptic Curve Cryptography
FL	Federated Learning
GDPR	General Data Regulation Protection
IIoT	Industrial Internet of Things
IoMT	Internet of Medical Things
IoT	Internet of Things
JWT	JSON Web Tokens
ML	Machine Learning
MTD	Moving Target Defense
OCSP	Online Certificate Status Protocol
PKI	Public Key Infrastructure
PoS	Proof of Stake
PoW	Proof of Work
RUL	Remaining Useful Lifetime
SSI	Self-Sovereign Identity
VC	Verifiable Credential
VP	Verifiable Presentation
W3C	World Wide Web Consortium

Bibliography

1. H. Abie and I. Balasingham. "Risk-based adaptive security for smart IoT in eHealth". In: *Proceedings of the 7th International Conference on Body Area Networks*. 2012, pp. 269–275.
2. A. Abraham, S. More, C. Rabensteiner, and F. Hörandner. "Revocable and Offline-Verifiable Self-Sovereign Identities". In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2020, pp. 1020–1027. DOI: 10.1109/TrustCom50675.2020.00136.
3. U. Ahmed, J. C.-W. Lin, and G. Srivastava. "Hyper-Graph Attention Based Federated Learning Method For Mental Health Detection". *IEEE Journal of Biomedical and Health Informatics*, 2022, pp. 1–1. DOI: 10.1109/JBHI.2022.3172269.
4. A. Alwarafy, K. A. Al-Thelaya, M. Abdallah, J. Schneider, and M. Hamdi. "A Survey on Security and Privacy Issues in Edge Computing-Assisted Internet of Things". *IEEE Internet of Things Journal*, 2020, pp. 1–1. DOI: 10.1109/JIOT.2020.3015432.
5. B. Alzahrani. "An Information-Centric Networking Based Registry for Decentralized Identifiers and Verifiable Credentials". *IEEE Access* 8, 2020, pp. 137198–137208. DOI: 10.1109/ACCESS.2020.3011656.
6. L. Ante, C. Fischer, and E. Strehle. "A bibliometric review of research on digital identity: Research streams, influential works and future research paths". *Journal of Manufacturing Systems* 62, 2022, pp. 523–538. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2022.01.005.
7. D. Apostolou, Y. Verginadis, and G. Mentzas. "In the Fog: Application Deployment for the Cloud Continuum". In: *2021 12th International Conference on In-*

formation, Intelligence, Systems Applications (IISA). 2021, pp. 1–7. DOI: 10.1109/IISA52424.2021.9555532.

8. H. F. Atlam, M. A. Azad, and N. F. Fadhel. "Efficient NFS Model for Risk Estimation in a Risk-Based Access Control Model". *Sensors* 22:5, 2022, p. 2005.

9. L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac. "A survey on IoT platforms: Communication, security, and privacy perspectives". *Computer Networks* 192, 2021, p. 108040. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2021.108040.

10. E. Battista, V. Casola, A. Mazzeo, and N. Mazzocca. "SIREN: a feasible moving target defence framework for securing resource-constrained embedded nodes". *International Journal of Critical Computer-Based Systems* 4:4, 2013. PMID: 59053, pp. 374–392. DOI: 10.1504/IJCCBS.2013.059053. eprint: https://www.inderscienceonline.com/doi/pdf/10.1504/IJCCBS.2013.059053. URL: https://www.inderscienceonline.com/doi/abs/10.1504/%20IJCCBS.2013.059053.

11. R. Beck. "Beyond Bitcoin: The Rise of Blockchain World". *Computer* 51:2, 2018, pp. 54–58. DOI: 10.1109/MC.2018.1451660.

12. S. Bhansali, A. Aris, A. Acar, H. Oz, and A. S. Uluagac. "A First Look at Code Obfuscation for WebAssembly". In: *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. WiSec '22. Association for Computing Machinery, San Antonio, TX, USA, 2022, pp. 140–145. ISBN: 9781450392167. DOI: 10.1145/3507657.3528560.

13. D. D. S. Braga, M. Niemann, B. Hellingrath, and F. B. D. L. Neto. "Survey on Computational Trust and Reputation Models". *ACM Comput. Surv.* 51:5, 2018. ISSN: 0360-0300. DOI: 10.1145/3236008.

14. L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz, et al. "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks". *Chainlink Labs*, 2021.

15. J. Bugeja, B. Vogel, A. Jacobsson, and R. Varshney. "IoTSM: An End-to-end Security Model for IoT Ecosystems". In: *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 2019, pp. 267–272. DOI: 10.1109/PERCOMW.2019.8730672.

16. G. Caldarelli. "Understanding the Blockchain Oracle Problem: A Call for Action". *Information* 11:11, 2020. ISSN: 2078-2489. DOI: 10.3390/info11110509.

17. M. Calvo and M. Beltrán. "A Model For risk-Based adaptive security controls". *Computers Security* 115, 2022, p. 102612. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2022.102612. URL: https://www.sciencedirect.com/science/article/pii/S0167404822000116.

18. P. Camacho and A. Hevia. "On the impossibility of batch update for cryptographic accumulators". In: *Progress in Cryptology–LATINCRYPT 2010: First International Conference on Cryptology and Information Security in Latin America, Puebla, Mexico, August 8-11, 2010, proceedings 1*. Springer. 2010, pp. 178–188.

19. J. Camenisch, M. Kohlweiss, and C. Soriente. "An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials". In: *Public Key Cryptography – PKC 2009*. Ed. by S. Jarecki and G. Tsudik. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 481–500. ISBN: 978-3-642-00468-1.

20. K. Cao, S. Hu, Y. Shi, A. W. Colombo, S. Karnouskos, and X. Li. "A Survey on Edge and Edge-Cloud Computing Assisted Cyber-Physical Systems". *IEEE Transactions on Industrial Informatics* 17:11, 2021, pp. 7806–7819. DOI: 10.1109/TII.2021.3073066.

21. V. Casola, A. D. Benedictis, C. Mazzocca, and R. Montanari. "Toward Automated Threat Modeling of Edge Computing Systems". In: *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*. 2021, pp. 135–140. DOI: 10.1109/CSR51186.2021.9527937.

22. V. Casola, A. De Benedictis, C. Mazzocca, and R. Montanari. "Designing Secure and Resilient Cyber-Physical Systems: a Model-based Moving Target Defense Approach". *IEEE Transactions on Emerging Topics in Computing*, 2022, pp. 1–12. DOI: 10.1109/TETC.2022.3197464.

23. V. Casola, A. De Benedictis, M. Rak, and U. Villano. "A Security SLA-Driven Moving Target Defense Framework to Secure Cloud Applications". In: *Proceedings of the 5th ACM Workshop on Moving Target Defense*. MTD '18. Association for Computing Machinery, Toronto, Canada, 2018, pp. 48–56. ISBN: 9781450360036. DOI: 10.1145/3268966.3268975. URL: https://doi.org/10.1145/3268966.3268975.

24. V. Casola, A. De Benedictis, M. Rak, and U. Villano. "Security-by-design in multi-cloud applications: An optimization approach". *Information Sciences* 454-455, 2018, pp. 344–362. ISSN: 0020-0255. DOI: 10.1016/j.ins.2018.04.081. URL: https://www.sciencedirect.com/science/article/pii/S0020025518303517.

25. M. Cebe and K. Akkaya. "Communication-efficient certificate revocation management for Advanced Metering Infrastructure and IoT Integration". *Future Generation Computer Systems* 115, 2021, pp. 267–278. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.08.023.

26. Z. B. Celik, P. McDaniel, and G. Tan. "Soteria: Automated IoT Safety and Security Analysis". In: *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 2018, pp. 147–158. ISBN: ISBN 978-1-939133-01-4. URL: https://www.usenix.org/conference/atc18/presentation/celik.

27. N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki. "Network Intrusion Detection for IoT Security Based on Learning Techniques". *IEEE Communications Surveys & Tutorials* 21:3, 2019, pp. 2671–2701. DOI: 10.1109/COMST.2019.2896380.

28. D. W. Chadwick, R. Laborde, A. Oglaza, R. Venant, S. Wazan, and M. Nijjar. "Improved Identity Management with Verifiable Credentials and FIDO". *IEEE Communications Standards Magazine* 3:4, 2019, pp. 14–20. DOI: 10.1109/MCOMSTD.001.1900020.

29. D. Choi, D. Kim, and S. Park. "A framework for context sensitive risk-based access control in medical information systems". *Computational and mathematical methods in medicine* 2015, 2015.

30. A. Chowdhary, S. Pisharody, and D. Huang. "SDN Based Scalable MTD Solution in Cloud Network". In: *Proceedings of the 2016 ACM Workshop on Moving Target Defense*. MTD '16. Association for Computing Machinery, Vienna, Austria, 2016, pp. 27–36. ISBN: 9781450345705. DOI: 10.1145/2995272.2995274. URL: https://doi.org/10.1145/2995272.2995274.

31. Cloud Security Alliance. "Top Threats to Cloud Computing: Egregious Eleven", 2019. URL: https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-egregious-eleven.

32. D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. *Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile*. Technical report. 2008.

33. D. C. Nguyen et al. "Federated learning meets blockchain in edge computing: Opportunities and challenges". *IEEE Internet of Things Journal*, 2021.

34. P. Derler, E. A. Lee, and A. Sangiovanni Vincentelli. "Modeling Cyber–Physical Systems". *Proceedings of the IEEE* 100:1, 2012, pp. 13–28. DOI: 10.1109/JPROC.2011.2160929.

35. P. Deutsch and J.-L. Gailly. *Zlib compressed data format specification version 3.3*. Technical report. 1996.

36. Docknetwork. "crypto-wasm-ts". URL: https://github.com/docknetwork/crypto-wasm-ts.

37. D. Dolev and A. Yao. "On the security of public key protocols". *IEEE Transactions on Information Theory* 29:2, 1983, pp. 198–208. DOI: 10.1109/TIT.1983.1056650.

38. T. J. Eggebraaten, J. W. Tenner, and J. C. Dubbels. "A health-care data model based on the HL7 Reference Information Model". *IBM Systems Journal* 46:1, 2007, pp. 5–18. DOI: 10.1147/sj.461.0005.

39. Y. C. Elloh Adja, B. Hammi, A. Serhrouchni, and S. Zeadally. "A blockchain-based certificate revocation management and status verification system". *Computers & Security* 104, 2021, p. 102209. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2021.102209.

40. Ethereum. *PROOF-OF-STAKE (POS)*. 2023. URL: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/.

41. EU. "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)", 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng.

42. B. Farahani, F. Firouzi, and M. Luecking. "The convergence of IoT and distributed ledger technologies (DLT): Opportunities, challenges, and solutions". *Journal of Network and Computer Applications* 177, 2021, p. 102936. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2020.102936.

43. M. Ficco, D. Granata, M. Rak, and G. Salzillo. "Threat Modeling of Edge-Based IoT Applications". In: *Quality of Information and Communications Technology*. Ed. by A. C. R. Paiva, A. R. Cavalli, P. Ventura Martins, and R. Pérez-Castillo. Springer International Publishing, Cham, 2021, pp. 282–296. ISBN: 978-3-030-85347-1.

44. N. Fotiou, V. A. Siris, G. C. Polyzos, Y. Kortesniemi, and D. Lagutin. "Capabilities-based access control for IoT devices using Verifiable Credentials". In: *2022 IEEE Security and Privacy Workshops (SPW)*. 2022, pp. 222–228. DOI: 10.1109/SPW54247.2022.9833873.

45. N. Fotiou, V. A. Siris, G. Xylomenos, and G. C. Polyzos. "IoT Group Membership Management Using Decentralized Identifiers and Verifiable Credentials, JOURNAL = Future Internet". 14:6, 2022. ISSN: 1999-5903. DOI: 10.3390/fi14060173.

46. I. Fundation. "IOTA Identity Framework". URL: https://github.com/iotaledger/identity.rs.

47. M. Ge, J. B. Hong, W. Guttmann, and D. S. Kim. "A framework for automating security analysis of the Internet of Things". *Journal of Network and Computer Applications* 83, 2017, pp. 12–27. DOI: https://doi.org/10.1016/j.jnca.2017.01.033.

48. B. C. Ghosh, S. Patranabis, D. Vinayagamurthy, V. Ramakrishna, K. Narayanam, and S. Chakraborty. "Private Certifier Intersection". *Cryptology ePrint Archive*, 2022.

49. M. T. Goodrich, R. Tamassia, and J. Hasić. "An efficient dynamic and distributed cryptographic accumulator". In: *Information Security: 5th International Conference, ISC 2002 Sao Paulo, Brazil, September 30–October 2, 2002 Proceedings 5*. Springer. 2002, pp. 372–388.

50. P. Gope and T. Hwang. "BSN-Care: A secure IoT-based modern healthcare system using body sensor network". *IEEE sensors journal* 16:5, 2015, pp. 1368–1376.

51. V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar. "A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures". *IEEE Access* 7, 2019, pp. 82721–82743. DOI: 10.1109/ACCESS.2019.2924045.

52. J. Höglund, M. Furuhed, and S. Raza. "Lightweight certificate revocation for low-power IoT with end-to-end security". *Journal of Information Security and Applications* 73, 2023, p. 103424. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2023.103424.

53. J. Höglund, S. Lindemer, M. Furuhed, and S. Raza. "PKI4IoT: Towards public key infrastructure for the Internet of Things". *Computers & Security* 89, 2020, p. 101658. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2019.101658.

54. C.-H. Hong and B. Varghese. "Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms". *ACM Comput. Surv.* 52:5, 2019. ISSN: 0360-0300. DOI: 10.1145/3326066. URL: https://doi.org/10.1145/3326066.

55. L. Hong, M. Luo, R. Wang, P. Lu, W. Lu, and L. Lu. "Big data in health care: Applications and challenges". *Data and information management* 2:3, 2018, pp. 175–197.

56. V. C. Hu, M. Iorga, W. Bao, A. Li, Q. Li, A. Gouglidis, et al. "General access control guidance for cloud systems". *NIST Special Publication 800-210*, 2020.

57. IHS Markit. *The Internet of Things: a movement, not a market*. Accessed: 2022-10-06. URL: %7Bhttps://cdn.ihs.com/www/pdf/IoT_ebook.pdf%7D.

58. K. Initaitive. "Identities of Things Discussion Group". URL: https://kantarainitiative.org/groups/idot/.

59. S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. 1st. Springer Publishing Company, Incorporated, 2011. ISBN: 9781461409762.

60. D. Koisser, P. Jauernig, G. Tsudik, and A.-R. Sadeghi. "V'CER: Efficient Certificate Validation in Constrained Networks". In: *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 2022, pp. 4491–4508. ISBN: 978-1-939133-31-1.

61. M. Y. Kubilay, M. S. Kiraz, and H. A. Mantar. "CertLedger: A new PKI model with Certificate Transparency based on blockchain". *Computers & Security* 85, 2019, pp. 333–352. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2019.05.013.

62. P. Kuwik, T. Largi, M. York, D. Crump, D. Livingston, and J. C. Squire. "The smart medical refrigerator". *IEEE Potentials* 24:1, 2005, pp. 42–45.

63. J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. "CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers". In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 539–556. DOI: 10.1109/SP.2017.17.

64. K. B. Letaief, Y. Shi, J. Lu, and J. Lu. "Edge Artificial Intelligence for 6G: Vision, Enabling Technologies, and Applications". *IEEE Journal on Selected Areas in Communications* 40:1, 2022, pp. 5–36. DOI: 10.1109/JSAC.2021.3126076.

65. *LG WebOS TV Developer*. Accessed: 2023-10-06. URL: %7Bhttps://webostv.developer.lge.com/%7D.

66. J. Li, Y. Bai, and N. Zaman. "A fuzzy modeling approach for risk-based access control in eHealth cloud". In: *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE. 2013, pp. 17–23.

67. Y. Li, Y. Chen, K. Zhu, C. Bai, and J. Zhang. "An Effective Federated Learning Verification Strategy and Its Applications for Fault Diagnosis in Industrial IoT Systems". *IEEE Internet of Things Journal* 9:18, 2022, pp. 16835–16849. DOI: 10.1109/JIOT.2022.3153343.

68. Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson. "An End-to-End Measurement of Certificate Revocation in the Web's PKI". In: *Proceedings of the 2015 Internet Measurement Conference*. IMC '15. Association for Computing Machinery, Tokyo, Japan, 2015, pp. 183–196. ISBN: 9781450338486. DOI: 10.1145/2815675.2815685.

69. Y. Liu, J. Wang, J. Li, S. Niu, and H. Song. "Machine Learning for the Detection and Identification of Internet of Things Devices: A Survey". *IEEE Internet of Things Journal* 9:1, 2022, pp. 298–320. DOI: 10.1109/JIOT.2021.3099028.

70. D. Maram, H. Malvai, F. Zhang, N. Jean-Louis, A. Frolov, T. Kell, T. Lobban, C. Moy, A. Juels, and A. Miller. "CanDID: Can-Do Decentralized Identity with Legacy Compatibility, Sybil-Resistance, and Accountability". In: *2021 IEEE Symposium on Security and Privacy (SP)*. 2021, pp. 1348–1366. DOI: 10.1109/SP40001.2021.00038.

71. C. Mazzocca, A. Acar, S. Uluagac, and R. Montanari. "EVOKE: Efficient Revocation of Verifiable Credentials in IoT Networks". In: *Under Review 33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, Philadelphia, PA, 2024.

72. C. Mazzocca, N. Romandini, M. Colajanni, and R. Montanari. "FRAMH: A Federated Learning Risk-Based Authorization Middleware for Healthcare". *IEEE Transactions on Computational Social Systems* 10:4, 2023, pp. 1679–1690. DOI: 10.1109/TCSS.2022.3210372.

73. C. Mazzocca, N. Romandini, M. Mendula, R. Montanari, and P. Bellavista. "TruFLaaS: Trustworthy Federated Learning as a Service". *IEEE Internet of Things Journal*, 2023, pp. 1–1. DOI: 10.1109/JIOT.2023.3282899.

74. C. Mazzocca, A. Sabbioni, R. Montanari, and M. Colajanni. "Evaluating Tangle Distributed Ledger for Access Control Policy Distribution in Multi-region Cloud Environments". In: *Quality of Information and Communications Technology*. Ed. by A. Vallecillo, J. Visser, and R. Pérez-Castillo. Springer International Publishing, Cham, 2022, pp. 296–306.

75. D. L. McGuinness, F. Van Harmelen, et al. "OWL web ontology language overview". *W3C recommendation* 10:10, 2004, p. 2004.

76. Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici. "N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders". *IEEE Pervasive Computing* 17:3, 2018, pp. 12–22.

77. Microsoft Corporation. "The STRIDE Threat Model", 2009. URL: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20).

78. M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma. "IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT". In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. 2017, pp. 2177–2184. DOI: 10.1109/ICDCS.2017.283.

79. D. Mills. "Internet time synchronization: the network time protocol". *IEEE Transactions on Communications* 39:10, 1991, pp. 1482–1493. DOI: 10.1109/26.103043.

80. S. Moreschini, F. Pecorelli, X. Li, S. Naz, D. Hästbacka, and D. Taibi. "Cloud Continuum: The Definition". *IEEE Access* 10, 2022, pp. 131876–131886. DOI: 10.1109/ACCESS.2022.3229185.

81. S. Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". *Decentralized business review*, 2008.

82. National Institute of Standards and Technology. "NIST Special Publication 800-53 Revision 4: Security and Privacy Controls for Federal Information Systems and Organizations", 2013. URL: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf.

83. R. E. Navas, F. Cuppens, N. Boulahia Cuppens, L. Toutain, and G. Z. Papadopoulos. "MTD, Where Art Thou? A Systematic Review of Moving Target Defense Techniques for IoT". *IEEE Internet of Things Journal* 8:10, 2021, pp. 7818–7832. DOI: 10.1109/JIOT.2020.3040358.

84. M. Nemec, M. Sys, P. Svenda, D. Klinec, and V. Matyas. "The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. Association for Computing Machinery, Dallas, Texas, USA, 2017, pp. 1631–1648. ISBN: 9781450349468. DOI: 10.1145/3133956.3133969. URL: https://doi.org/10.1145/3133956.3133969.

85. D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor. "Federated Learning Meets Blockchain in Edge Computing: Opportunities and Challenges". *IEEE Internet of Things Journal* 8:16, 2021, pp. 12806–12825. DOI: 10.1109/JIOT.2021.3072611.

86. NIST. *Glossary - Cyber resiliency*. 2018. URL: https://csrc.nist.gov/glossary/term/cyber%5C_resiliency.

87. NIST. *NIST Special Publication 800-160, Volume 2, Revision 1. Developing Cyber-Resilient Systems: A Systems Security Engineering Approach*. 2021. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/%20NIST.SP.800-160v2r1.pdf.

88. S. M. Nóbrega Gonçalves, A. Tomasi, A. Bisegna, G. Pellizzari, and S. Ranise. "Verifiable Contracting: A Use Case for Onboarding and Contract Offering in Financial Services with eIDAS and Verifiable Credentials". In: *Computer Security: ESORICS 2020 International Workshops, DETIPS, DeSECSys, MPS, and SPOSE, Guildford, UK, September 17–18, 2020, Revised Selected Papers 25*. Springer. 2020, pp. 133–144.

89. OWASP. "The Ten Most Critical Web Application Security Risks", 2017.

90. R. P. Parameswarath, P. Gope, and B. Sikdar. "A Privacy-Preserving Authenticated Key Exchange Protocol for V2G Communications Using SSI". *IEEE Transactions on Vehicular Technology*, 2023, pp. 1–16. DOI: 10.1109/TVT.2023.3281371.

91. S. Popov. "The tangle". *White paper* 1:3, 2018, p. 30.

92. J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang. "A Survey on Access Control in the Age of Internet of Things". *IEEE Internet of Things Journal* 7:6, 2020, pp. 4682–4696. DOI: 10.1109/JIOT.2020.2969326.

93. W. Recommendation. "Revocation List 2020", 2021. URL: https://w3c-ccg.github.io/vc-status-rl-2020/.

94. M. H. u. Rehman, A. M. Dirir, K. Salah, E. Damiani, and D. Svetinovic. "TrustFed: A Framework for Fair and Trustworthy Cross-Device Federated Learning in IIoT". *IEEE Transactions on Industrial Informatics* 17:12, 2021, pp. 8485–8494. DOI: 10.1109/TII.2021.3075706.

95. D. Rosendo, A. Costan, P. Valduriez, and G. Antoniu. "Distributed intelligence on the Edge-to-Cloud Continuum: A systematic literature review". *Journal of Parallel and Distributed Computing* 166, 2022, pp. 71–94. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2022.04.004.

96. S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. *X.509 internet public key infrastructure online certificate status protocol-OCSP*. Technical report. 2013.

97. A. Saxena, K. Goebel, D. Simon, and N. Eklund. "Damage propagation modeling for aircraft engine run-to-failure simulation". In: *2008 International Conference on Prognostics and Health Management*. 2008, pp. 1–9. DOI: 10.1109/PHM.2008.4711414.

98. M. Serror, S. Hack, M. Henze, M. Schuba, and K. Wehrle. "Challenges and Opportunities in Securing the Industrial Internet of Things". *IEEE Transactions on Industrial Informatics* 17:5, 2021, pp. 2985–2996. DOI: 10.1109/TII.2020.3023507.

99. K. Sha, R. Errabelly, W. Wei, T. A. Yang, and Z. Wang. "EdgeSec: Design of an Edge Layer Security Service to Enhance IoT Security". In: *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*. 2017, pp. 81–88. DOI: 10.1109/ICFEC.2017.7.

100. K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim. "Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios". *IEEE Access* 8, 2020, pp. 23022–23040. DOI: 10.1109/ACCESS.2020.2970118.

101. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. "Edge Computing: Vision and Challenges". *IEEE Internet of Things Journal* 3:5, 2016, pp. 637–646. DOI: 10.1109/JIOT.2016.2579198.

102. W. Shi and S. Dustdar. "The Promise of Edge Computing". *Computer* 49:5, 2016, pp. 78–81. DOI: 10.1109/MC.2016.145.

103. A. Singla and E. Bertino. "Blockchain-Based PKI Solutions for IoT". In: *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*. 2018, pp. 9–15. DOI: 10.1109/CIC.2018.00-45.

104. L. Sion, K. Yskout, D. Van Landuyt, and W. Joosen. "Risk-Based Design Security Analysis". In: *Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment*. SEAD '18. Association for Computing Machinery, Gothenburg, Sweden, 2018, pp. 11–18. ISBN: 9781450357272. DOI: 10.1145/3194707.3194710. URL: https://doi.org/10.1145/3194707.3194710.

105. T. Smith, L. Dickinson, and K. Seamons. "Let's Revoke: Scalable Global Certificate Revocation". *Network and Distributed Systems Security (NDSS) Symposium 2020*. DOI: 10.14722/ndss.2020.24084.

106. J. So, B. Güler, and A. S. Avestimehr. "Byzantine-Resilient Secure Federated Learning". *IEEE Journal on Selected Areas in Communications* 39:7, 2021, pp. 2168–2181. DOI: 10.1109/JSAC.2020.3041404.

107. K. Srivastava and N. Shekokar. "Machine learning based risk-adaptive access control system to identify genuineness of the requester". In: *Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough*. Springer, 2020, pp. 129–143.

108. SSL Labs. "SSL Threat Model", 2018. URL: https://www.ssllabs.com/projects/ssl-threat-model/.

109. Standard OASIS. "extensible access Control markup language (xacml) version 3.0". *A:(22 January 2013). URl: http://docs. oasis-open. org/xacml/3.0/xacml-3.0-core-spec-os-en. html*, 2013. URL: http://docs.%20oasis-open.%20org/xacml/3.0/xacml-3.0-core-spec-os-en.%20html.

110. C. Sun and L. Zhang. "Design and Modeling of Intelligent Home Security Monitoring System Based on CPS". In: *2021 IEEE 12th International Conference on Software Engineering and Service Science (ICSESS)*. 2021, pp. 186–189. DOI: 10.1109/ICSESS52187.2021.9522189.

111. P. Tedeschi and S. Sciancalepore. "Edge and Fog Computing in Critical Infrastructures: Analysis, Security Threats, and Research Challenges". In: *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. 2019, pp. 1–10. DOI: 10.1109/EuroSPW.2019.00007.

112. E. Tekiner, A. Acar, and A. S. Uluagac. "A lightweight IoT cryptojacking detection mechanism in heterogeneous smart home networks". In: *Proc. of the ISOC Network and Distributed System Security Symposium (NDSS)*. 2022.

113. A. Tesei, L. Di Mauro, M. Falcitelli, S. Noto, and P. Pagano. "IOTA-VPKI: A DLT-Based and Resource Efficient Vehicular Public Key Infrastructure". In: *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. 2018, pp. 1–6. DOI: 10.1109/VTCFall.2018.8690769.

114. M. Torquato and M. Vieira. "Moving target defense in cloud computing: A systematic mapping study". *Computers & Security* 92, 2020, p. 101742. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2020.101742. URL: https://www.sciencedirect.com/science/article/pii/%20S0167404820300286.

115. G. Vitto and A. Biryukov. "Dynamic universal accumulator with batch update over bilinear groups". In: *Topics in Cryptology–CT-RSA 2022: Cryptographers' Track at the RSA Conference 2022, Virtual Event, March 1–2, 2022, Proceedings*. Springer. 2022, pp. 395–426.

116. L. Wang, Y. Tian, and D. Zhang. "Toward Cross-Domain Dynamic Accumulator Authentication Based on Blockchain in Internet of Things". *IEEE Transactions on Industrial Informatics* 18:4, 2022, pp. 2858–2867. DOI: 10.1109/TII.2021.3116049.

117. S. Wang, H. Li, J. Chen, J. Wang, and Y. Deng. "DAG blockchain-based lightweight authentication and authorization scheme for IoT devices". *Journal of Information Security and Applications* 66, 2022, p. 103134. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2022.103134.

118. T. Wang, Q. Wang, Z. Shen, Z. Jia, and Z. Shao. "Understanding Characteristics and System Implications of DAG-Based Blockchain in IoT Environments". *IEEE Internet of Things Journal* 9:16, 2022, pp. 14478–14489. DOI: 10.1109/JIOT.2021.3108527.

119. C. Watson and T. Zaw. *OWASP Automated Threat Handbook*. 2018. URL: https://github.com/OWASP/www-project-automated-threats-to-web-applications/blob/master/assets/files/EN/automated-threat-handbook-EN-1v20.pdf.

120. T. J. Williams. "The Purdue enterprise reference architecture". *Computers in Industry* 24:2, 1994, pp. 141–158. ISSN: 0166-3615. DOI: https://doi.org/10.1016/0166-3615(94)90017-5.

121. Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv. "Edge Computing Security: State of the Art and Challenges". *Proceedings of the IEEE* 107:8, 2019, pp. 1608–1631. DOI: 10.1109/JPROC.2019.2918437.

122. Y. Yang, X. Zheng, W. Guo, X. Liu, and V. Chang. "Privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system". *Information Sciences* 479, 2019, pp. 567–592.

123. Y. Yuehong, Y. Zeng, X. Chen, and Y. Fan. "The internet of things in healthcare: An overview". *Journal of Industrial Information Integration* 1, 2016, pp. 3–13.

124. B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga. "A survey of intrusion detection in Internet of Things". *Journal of Network and Computer Applications* 84, 2017, pp. 25–37. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2017.02.009.

125. J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu. "Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues". *IEEE Access* 6, 2018, pp. 18209–18237. DOI: 10.1109/ACCESS.2018.2820162.

126. Q. Zhu, S. W. Loke, R. Trujillo-Rasua, F. Jiang, and Y. Xiang. "Applications of Distributed Ledger Technologies to the Internet of Things: A Survey". *ACM Comput. Surv.* 52:6, 2019. ISSN: 0360-0300. DOI: 10.1145/3359982.