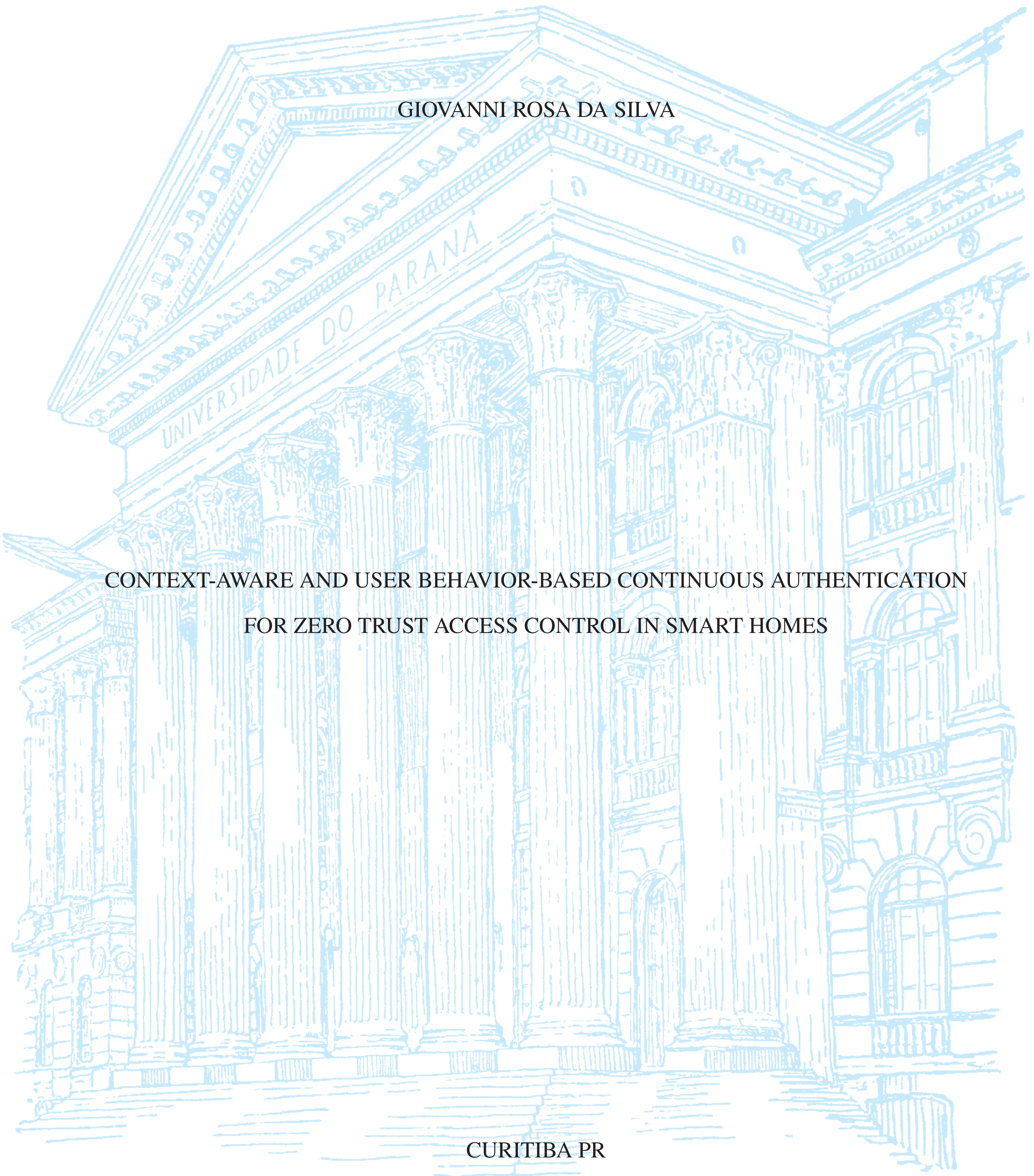
UNIVERSIDADE FEDERAL DO PARANÁ



GIOVANNI ROSA DA SILVA

CONTEXT-AWARE AND USER BEHAVIOR-BASED CONTINUOUS AUTHENTICATION FOR ZERO TRUST ACCESS CONTROL IN SMART HOMES

CURITIBA PR

2023

GIOVANNI ROSA DA SILVA

CONTEXT-AWARE AND USER BEHAVIOR-BASED CONTINUOUS AUTHENTICATION FOR ZERO TRUST ACCESS CONTROL IN SMART HOMES

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Aldri Luiz dos Santos.

CURITIBA PR

2023

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA CIÊNCIA E TECNOLOGIA

Silva, Giovanni Rosa da
Context-aware and user behavior-based continuous authentication for zero trust access control in smart homes. / Giovanni Rosa da Silva. – Curitiba, 2023.
1 recurso on-line : PDF.

Dissertação (Mestrado) - Universidade Federal do Paraná, Setor de Ciência Exatas, Programa de Pós-Graduação em Informática.

Orientador: Aldri Luiz dos Santos.

1. Internet das Coisas. 2. Autenticação contínua. 3. Comportamento humano. 4. Habitações (Inteligentes). I. Santos, Aldri Luiz dos. III. Universidade Federal do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

Bibliotecária: Roseny Rivelini Morciani CRB-9/1585

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **GIOVANNI ROSA DA SILVA** intitulada: **Context-Aware and User Behavior-Based Continuous Authentication for Zero Trust Access Control in Smart Homes**, sob orientação do Prof. Dr. ALDRI LUIZ DOS SANTOS, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 24 de Fevereiro de 2023.

Assinatura Eletrônica
24/02/2023 13:00:11.0
ALDRI LUIZ DOS SANTOS
Presidente da Banca Examinadora

Assinatura Eletrônica
01/03/2023 10:48:12.0
MICHELE NOGUEIRA LIMA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
27/02/2023 08:46:33.0
LUIZ HENRIQUE ANDRADE CORREIA
Avaliador Externo (UNIVERSIDADE FEDERAL DE LAVRAS)

Dedico este trabalho ao meu avô Darci Rosa da Silva (in memoriam) e minha avós Stella Maris Godoe Borges (in memoriam) e Deolinda Machado da Silva

ACKNOWLEDGEMENTS

A Deus, pela minha vida, e por me permitir ultrapassar todos os obstáculos encontrados ao longo da realização deste trabalho.

À minha esposa Roberta Samistraro Tomigian por ser paciente e um suporte nos momentos de maior dificuldade. Nos períodos de angústia esteve ao meu lado e me incentivou a nunca desistir. Abriu mão de muitas ocasiões para estar ao meu lado e entender que precisava de estágios de dedicação.

Aos meus pais, Amauri Rosa da Silva e Silvia Adriana Borges, e família, por me proporcionar amor, incentivo, e uma educação de qualidade que me permitiu buscar a realização dos meus sonhos. Nunca deixaram de enfatizar que a educação é o caminho para evoluirmos.

Ao meu orientador Aldri Luiz dos Santos, que conduziu o trabalho com paciência e dedicação, sempre disponível a compartilhar todo o seu vasto conhecimento. Com sua ajuda evolui e amadureci meu método de pesquisa e escrita, além de aprender em todos nossos encontros questões sociais e filosóficas que levarei comigo para sempre.

Aos colegas do laboratório do Núcleo de Redes Sem Fio e Redes Avançadas (NR2), que me receberam com carinho e ajudaram no processo de construção deste trabalho. Em especial ao Agnaldo de Souza Batista que me acompanhou desde a graduação e com competência me ajudou em momentos de dificuldade. Ao Carlos Alberto Pedroso, Yan Uehara de Moraes, Alisson Yury Dutra de Medeiros, e José Everaldo Andrade Junior pelas contribuições ao meu trabalho com paciência de gerar críticas construtivas que tornaram a pesquisa mais robusta e relevante.

Ao curso de Ciência da Computação do Departamento de Informática (DInf) da Universidade Federal do Paraná (UFPR), essencial no meu processo de formação acadêmica, pela paciência e excelência, e por tudo o que aprendi ao longo dos anos na graduação. Ao Programa de Pós-Graduação em Informática (PPGInf) da Universidade Federal do Paraná (UFPR), pelo suporte ao meu trabalho com infraestrutura e ensino na linha de pesquisa de Redes e Sistemas Distribuídos. Agradeço também aos funcionários do departamento pelo auxílio com processos burocráticos da instituição.

À CINQ Technologies e ao Marcelo Wzorek, essenciais no meu processo de formação profissional, pela compreensão em me permitir trabalhar meio período para que pudesse me dedicar a este trabalho. Proporcionaram-me a capacidade de concluir essa pesquisa com tranquilidade enquanto continuei independente financeiramente e avançando no espectro profissional.

À Sociedade Brasileira de Computação (SBC), a qual sou associado desde 2019, por trabalhar há mais de 45 anos pelo avanço científico e tecnológico no Brasil. Com seus mais de 4 mil membros ativos proporciona um ambiente propício à aquisição de conhecimento, informação e qualificação profissional.

Ao Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg), por me permitir a publicação de parte da pesquisa e me proporcionar uma experiência única para validação do trabalho e aquisição de conhecimento sobre a área de segurança.

A todos os pesquisadores e comunidades de computação, que me permitiram, sobre o ombro de gigantes, realizar minha pesquisa a partir de um ponto tão avançado na história da humanidade. Pela dedicação de milhares de pessoas minha pesquisa foi possível.

E a todos aqueles que contribuíram, de alguma forma, para a realização deste trabalho.

RESUMO

Embora as casas inteligentes tenham se tornado populares recentemente, as pessoas ainda estão muito preocupadas com questões de segurança, proteção e privacidade. Estudos revelaram que questões de privacidade das pessoas geram prejuízos fisiológicos e financeiros porque as casas inteligentes são ambientes de convivência íntima. Além disso, nossa pesquisa revelou que os ataques de impersonificação são uma das ameaças mais graves contra casas inteligentes porque comprometem a confidencialidade, autenticidade, integridade e não repúdio. Normalmente, abordagens para construir segurança para Sistemas de Casas Inteligentes (SHS) requerem dados históricos para implementar controle de acesso e Sistemas de Detecção de Intrusão (IDS), uma vulnerabilidade à privacidade dos habitantes. Além disso, a maioria dos trabalhos depende de computação em nuvem ou recursos na nuvem para executar tarefas de segurança, que os invasores podem atacar para atingir a confidencialidade, integridade e disponibilidade. Além disso, os pesquisadores não consideram o uso indevido de SHS ao forçar os usuários a interagir com os dispositivos por meio de seus smartphones ou tablets, pois eles costumam interagir por qualquer meio, como assistentes virtuais e os próprios dispositivos. Portanto, os requisitos do sistema de segurança para residências inteligentes devem compreender percepção de privacidade, resposta de baixa latência, localidade espacial e temporal, extensibilidade de dispositivo, proteção contra impersonificação, isolamento de dispositivo, garantia de controle de acesso e levar em consideração a verificação atualizada com um sistema confiável. Para atender a esses requisitos, propomos o sistema ZASH (*Zero-Aware Smart Home*) para fornecer controle de acesso para as ações do usuário em dispositivos em casas inteligentes. Em contraste com os trabalhos atuais, ele aproveita a autenticação contínua com o paradigma de Confiança Zero suportado por ontologias configuradas, contexto em tempo real e atividade do usuário. A computação de borda e a Cadeia de Markov permitem que o ZASH evite e mitigue ataques de impersonificação que visam comprometer a segurança dos usuários. O sistema depende apenas de recursos dentro de casa, é autossuficiente e está menos exposto à exploração externa. Além disso, funciona desde o dia zero sem a exigência de dados históricos, embora conte com o passar do tempo para monitorar o comportamento dos usuários. O ZASH exige prova de identidade para que os usuários confirmem sua autenticidade por meio de características fortes da classe *Something You Are*. O sistema executa o controle de acesso nos dispositivos inteligentes, portanto, não depende de intermediários e considera qualquer interação usuário-dispositivo. A princípio, um teste inicial de algoritmos com um conjunto de dados sintético demonstrou a capacidade do sistema de se adaptar dinamicamente aos comportamentos de novos usuários, bloqueando ataques de impersonificação. Por fim, implementamos o ZASH no simulador de rede ns-3 e analisamos sua robustez, eficiência, extensibilidade e desempenho. De acordo com nossa análise, ele protege a privacidade dos usuários, responde rapidamente (cerca de 4,16 ms), lida com a adição e remoção de dispositivos, bloqueia a maioria dos ataques de impersonificação (até 99% com uma configuração adequada), isola dispositivos inteligentes e garante o controle de acesso para todas as interações.

Palavras-chave: Internet das Coisas; Casas Inteligentes; Controle de Acesso; Autenticação Contínua; Confiança Zero; Sensível ao Contexto; Baseado no Comportamento do usuário.

ABSTRACT

Although smart homes have become popular recently, people are still highly concerned about security, safety, and privacy issues. Studies revealed that issues in people's privacy generate physiological and financial harm because smart homes are intimate living environments. Further, our research disclosed that impersonation attacks are one of the most severe threats against smart homes because they compromise confidentiality, authenticity, integrity, and non-repudiation. Typically, approaches to build security for Smart Home Systems (SHS) require historical data to implement access control and Intrusion Detection Systems (IDS), a vulnerability to the inhabitant's privacy. Additionally, most works rely on cloud computing or resources in the cloud to perform security tasks, which attackers can exploit to target confidentiality, integrity, and availability. Moreover, researchers do not regard the misuse of SHS by forcing users to interact with devices through their smartphones or tablets, as they usually interact by any means, like virtual assistants and devices themselves. Therefore, the security system requirements for smart homes should comprehend privacy perception, low latency in response, spatial and temporal locality, device extensibility, protection against impersonation, device isolation, access control enforcement, and taking into account the refresh verification with a trustworthy system. To attend to those requirements, we propose the ZASH (Zero-Aware Smart Home) system to provide access control for the user's actions on smart devices in smart homes. In contrast to current works, it leverages continuous authentication with the Zero Trust paradigm supported by configured ontologies, real-time context, and user activity. Edge computing and Markov Chain enable ZASH to prevent and mitigate impersonation attacks that aim to compromise users' security. The system relies only on resources inside the house, is self-sufficient, and is less exposed to outside exploitation. Furthermore, it works from day zero without the requirement of historical data, though it counts on that as time passes to monitor the users' behavior. ZASH requires proof of identity for users to confirm their authenticity through strong features of the *Something You Are* class. The system enforces access control in smart devices, so it does not depend on intermediaries and considers any user-device interaction. At first, an initial test of algorithms with a synthetic dataset demonstrated the system's capability to dynamically adapt to new users' behaviors withal blocking impersonation attacks. Finally, we implemented ZASH in the ns-3 network simulator and analyzed its robustness, efficiency, extensibility, and performance. According to our analysis, it protects users' privacy, responds quickly (around 4.16 ms), copes with adding and removing devices, blocks most impersonation attacks (up to 99% with a proper configuration), isolates smart devices, and enforces access control for all interactions.

Keywords: Internet of Things; Smart Home; Access Control; Continuous Authentication; Zero Trust; Context-Aware; User Behavior-Based.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

5G	Cellular Network Fifth Generation Technology
6G	Cellular Network Sixth Generation Technology
6LowPAN	IPv6 over Low power Wireless Personal Area Networks
ABA	Anomaly-based Behavior Analysis
ABAC	Attribute-Based Access Control
AC	Access Control
ACD	Access Control Distance
ACE	Access Control Enforcement
ACL	Access Control List
ACRT	Access Control Response Time
AD	Activity Discovery
AD	Anomaly Detection
ADR	Attacks Denied Rate
AF	Activity Fail
ANS	Advanced Networks and Services
ARPA	Advanced Research Projects Agency
ARPANET	Advanced Research Projects Agency Network
BAN	Body Area Network
CA	Continuous Authentication
CAAC	Context-Aware Access Control
CapBAC	Capability-Based Access Control
CF	Context Fail
CIA	Confidentiality, Integrity, and Availability
CL	Capabilities List
CoAP	Constrained Application Protocol
CPS	Cyber-Physical Systems
CPSS	Cyber-Physical Social Systems
DAC	Discretionary Access Control
DBMS	Database Management System
DE	Device Extensibility
DI	Device Isolation
DINF	Departamento de Informática (Department of Informatics)
DoS	Denial of Service
DRM	Digital Right Management

DSL	Digital Subscriber Line
DT	Decision Tree
ECU	Electronic Control Units
ED	Episode Discovery
GDPR	General Data Protection Regulation
GMM	Gaussian Mixture Model
GPS	Global Positioning System
GSM	Global System for Mobile
HAN	Home Area Network
HIDS	Host-based Intrusion Detection System
HMM	Hidden Markov Model
IaaS	Infrastructure as a Service
IB	Instance-Based Learner
ICM	Information City Model
ICS	Industrial Control Systems
ID	Identification
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IMD	Implantable Medical Device
IoE	Internet of Everything
IoP	Internet of People
IoT	Internet of Things
IP	Internet Protocol
KNN	K Nearest Neighbor
LAN	Local Area Network
LGPD	Lei Geral de Proteção de Dados (General Data Protection Law)
LISP	Locator ID Separation Protocol
LPWAN	Low Power Wide Area Network
M2M	Machine-to-Machine
MAC	Media Access Control
MAC	Mandatory Access Control
MEC	Mobile Edge Computing
MIDS	Mixed IDS
MITM	Man-in-the-Middle
ML	Machine Learning
MLP	Multilayer Perceptron
MLT	Logistic Model Tree
MQTT	Message Queuing Telemetry Transport

NAP	Network Access Points
NBA	Network Behavior Analysis
NFC	Near Field Communication
NIDS	Network-based Intrusion Detection System
NN	Neural Network
NoSQL	Not Only Structured Query Language
NR2	Wireless Networks and Advanced Networks Core
NSF	National Science Foundation
NSFNET	National Science Foundation Network
OpenSHS	Open Smart Home Simulator
OrBAC	Organization-Based Access Control
ORM	Object Role Modelling
OF	Ontology Fail
P2M	People-to-Machine
P2P	People-to-People
PaaS	Platform as a Service
PAN	Personal Area Network
PCAP	Packet Capture
PDML	Packet Description Markup Language
PDP	Policy Decision Point
PE	Policy Engine
PEP	Policy Enforcement Point
PIP	Policy Information Point
PIR	Passive Infrared
PPGINF	Programa de Pós-Graduação em Informática (Graduate Program in Informatics)
PR	Privacy Risk
QoC	Quality of Context Information
QoS	Quality of Service
RBAC	Role-Based Access Control
RD	Requests Denied
RFID	Radio-Frequency Identification
RG	Requests Granted
RPL	IPv6 Routing Protocol for Low Power and Lossy Networks
SaaS	Software as a Service
SBSeg2021	XXI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais 2021 (XXI Brazilian Symposium on Information Security and Computer Systems 2021)

SC	Sequence Classification
SHS	Smart Home Systems
SIMADL	Simulated Activities of Daily Living Dataset
SQL	Structured Query Language
SRI	Stanford Research Institute
SSO	Single Sign-On
STL	Spatial and Temporal Locality
SVM	Support Vector Machine
SWRL	Semantic Web Rule Language
SYN	Synchronous
TA	Trust Algorithm
TBAC	Trust-Based Access Control
TCP	Transmission Control Protocol
UBA	User Behavior Analytics
UCLA	University of California at Los Angeles
UCON	Usage Control
UCSB	University of California at Santa Barbara
UFPR	Universidade Federal do Paraná (Federal University of Paraná)
UML	Unified Modelling Language
USD	United States Dollars
UTAH	University of Utah
VLAN	Virtual Local Area Network
Wi-Fi	Wireless Fidelity
WIDS	Wireless-based Intrusion Detection System
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
WWW	World Wide Web
ZASH	Zero-Aware Smart Home
ZT	Zero-Trust
ZTA	Zero Trust Architecture

CONTENTS

1 INTRODUCTION

The Internet of Things (IoT) is already an established paradigm where many devices (things) are interconnected and provide various services to people. As things are connected, many created systems have assumed social characteristics intrinsic to human problems. Thus, Cyber-Physical Social Systems (CPSS) are emerging (Dong and Ansari, 2020), evolving towards the Internet of Everything (IoE), which includes people, data, and things in a unified process (de Matos et al., 2017). IoE aims to connect anything with Internet-connection capability, from people to TVs, cars, and sensors, which can be reached through billions and trillions of connections. Besides, it enables an evolution in many applications, such as transport and logistics, healthcare, and smart environments (smart homes, smart cities). Hence, CPSS and IoE are intimately linked by human factors that bring human knowledge, mental capacity, and sociocultural elements using things to mix the physical world with cyberspace (Dong and Ansari, 2020).

Remarkably, smart home settings have been gaining popularity as there are more Internet users, more importance on green energy, more concern for security, and more smart gadgets available to more people (MarketsAndMarkets, 2023). Recently due to the pandemic of COVID-19 people mandatorily needed to stay at home (McQuigge, 2020), increasing its importance in daily life. Naturally, people will look for more comfort, security, and safety in their homes when they need to spend more time there. However, people still fear cybersecurity vulnerabilities that can expose personal information and their families' physical safety. Lee (2020) revealed that the resistance to smart home adoption comes from vulnerabilities in technology, with fragile security, nonexistent or weak laws on digital crimes, unreliable service providers, and the users themselves, due to misuse or inexperience.

Since Smart Home Systems (SHS) comprehend CPSS, the physical security measures in smart homes are as important as protecting people's data from digital threats. For instance, impersonation attacks are common in SHS once they allow attackers to steal and use authentic identities, compromising confidentiality, integrity, authenticity, and non-repudiation (Mocrii et al., 2018). Therefore, these attacks often involve social engineering, when attackers deceive victims and steal their credentials; and eavesdropping, when attackers steal information on the fly (Humayed et al., 2017). After that, the attacker can use someone's credentials to gain access to SHS and perform malicious actions to monitor house activity further or even invade the property, putting the resident's data and physical privacy at risk. Reig et al. (2021) found that data collection and monitoring, especially when undisclosed, was the most significant concern about the advancement of smart home technology.

Prior works on SHS security focus on access control, continuous authentication, and Intrusion Detection System (IDS). Furthermore, many of them use context-based decisions (Ashibani et al., 2019; Ghosh et al., 2019; Sikder et al., 2019a) and behavior-based decisions (Ashibani and Mahmoud, 2019; Ghosh et al., 2019; Amraoui et al., 2020). Although few are suitable to endure impersonation attacks (Ashibani et al., 2019; Ghosh et al., 2019; Sikder et al., 2019a), they present some issues as the dependence on cloud computing, external services, and historical data to infer information about the environment or user activities, leading to security and privacy breaches. Another issue refers to the dependence on end-user devices to enforce access control as they have a general purpose, being more exposed to exploitation. Additionally, it can lead to vulnerability due to misuse once users interact with SHS by different means, like smart assistants and devices. Therefore, the state-of-the-art security systems for smart homes revealed the requirements for smart homes security systems: privacy perception, protect the data and physical privacy of the inhabitants; low latency in response, control of the devices in real-time; spatial and temporal locality, as the system must be self-sufficient; device extensibility, support addition and removal of devices; protection against impersonation, prevent, detect and mitigate impersonation attacks; device isolation, establish appropriate access privilege levels to devices; and access control enforcement, support digital and physical user-device interaction.

1.1 MOTIVATION

Smart homes are popular in developed countries due to the broad access to smart devices and gaining more importance worldwide as technology spreads and becomes accessible. The increasing number of Internet users and the growing adoption of smart devices have contributed to the growth of the overall IoT market recently and for the following years. The interest of consumers in video doorbells, voice-assisted technologies (such as Alexa and Google Home), surveillance systems, and home automation explains the expected increase in the smart homes market from USD 101.7 billion in 2023 to USD 163.7 billion by 2028 (MarketsAndMarkets, 2023).

Although the smart home's interest is high, the resistance to its adoption remains on security issues (Lee, 2020). People do not trust the provider companies, dreading unauthorized personal data collection and weak security measures. For example, Amazon and Google require third-party partner companies to continually share status updates with them, potentially exposing user data to attacks. Previously, access to this information occurred only upon issuing a command (Carlsen, 2021). They also fear technology, with fragile security systems. The law is sometimes nonexistent or weak regarding digital crimes. The recently approved GDPR in Europe and LGPD in Brazil are legislations to regulate sensitive personal data use and impose sanctions on entities that misuse or do not store people's data securely (Intersoft Consulting, 2019). Besides, most people do not employ the best security practices regarding their digital devices and accounts. In the research Yubico (2019) about human behavior related to passwords, 69% of the respondents admitted to sharing them with colleagues, 51% reuse them among accounts, 57% of those who suffered phishing attacks did not change their behaviors, 67% do not use two-factor authentication in their accounts, and 55% do not use at work. Lastly, 57% preferred an authentication method that does not involve passwords. Ultimately, most potential smart home consumers are inexperienced with smart devices, which could lead to misuse.

As IoT spreads inside people's homes, the vulnerabilities inherent to its technologies pose threats to users. There could be financial loss, breaches of data privacy, or loss of control of computer devices in cyberspaces applications. The smart home's security issues may cause consequences in cyberspace and the physical world, whereby occupants' security, privacy, safety, and well-being are threatened (Heartfield et al., 2018). For example, smart meter transmission breaches can expose the inhabitants' presence or consumption pattern. Researches revealed the vulnerability of common smart devices, like baby monitor cameras, where attackers can visually spy on the house (Albrecht and Mcintyre, 2015; BBC, 2020). Attackers can control home devices to compromise people's safety, such as opening smart locks, turning off lights, or even heating systems. There are also cyber impacts, like confidentiality when attackers gain unauthorized access to information; integrity when attackers compromise devices to use for large-scale phishing attacks; availability when users cannot control their devices; and non-repudiation when attacker camouflages their traces. Heartfield et al. (2018) also exposed emotional consequences on inhabitants caused by smart home attacks, like high-stress levels, discomfort, and a personal sense of losing control and privacy.

The smart homes can be attacked externally by an agent outside the SHS and internally by an authentic user. Mocrii et al. (2018) described that the most common threats in a smart home are: eavesdropping, when attackers obtain valuable information and can use it to plan a further active or physical attack; software exploitation, which allows cybercriminals to exploit devices vulnerabilities; denial of service, which impacts the capability of users to control their homes, generally, through the Internet; ransomware, when attackers encrypt users information and ask for a ransom to release secret key; and impersonation, when an attacker tries to act on behalf of the legitimate user. Few works on smart homes focus on securing inhabitants against insider attacks. Ashibani et al. (2019) proposed a continuous authentication with a context-based decision by collecting instant contextual information to evaluate the trust to assign the user security level and access threshold. The higher the confidence achieved by available parameters (e.g., username, password, Bluetooth, and Wi-Fi proximity), the higher the service privilege user can access. Ghosh et al. (2019) exposed an authorization framework with context and behavior-based decisions by modeling belief on the user's usage history and on contextual factors (e.g., location of the requester, time of request). Sikder et al. (2019a) used a Markov Chain to control whether a request is normal or anomalous to detect malicious activity and, in a later work, Sikder et al. (2020) proved that Markov Chain is the best technique to detect anomaly requests in a context-aware environment.

1.2 PROBLEM

Impersonation attacks in SHS consist of several steps and consequences. First, the attacker steals any people's credentials, usually, those with more privileges, using social engineering or an eavesdropping attack. According to Krombholz et al. (2015), the social engineering attack vectors can be classified as phishing, the attempt to acquire sensitive information by manipulating people by masquerading as a trustworthy entity in an electronic communication medium; dumpster diving, as the practice to dig through the trash of people to find sensitive information; shoulder surfing, when attacker get information by direct observation without people's consent; reverse social engineering, when attackers create a situation in which the victim needs help from them to receive privileged information; waterholing, which uses a compromised website that is likely to be attractive to the chosen victim; advanced persistent threat, refers to long-term strategies conducted by an attacker who wants to compromise a system persistently; and baiting, when the victim accesses a malware-infected storage medium left by the attacker. Attackers can alternatively benefit from a man-in-the-middle attack to access the SHS. After that, they can monitor the house activity from inside to plan a physical invasion, steal more privileged information and even control the home devices to compromise inhabitants' safety.

Masquerading relates to another usual term when an attacker with stolen credentials from authentic user access the SHS (Komninos et al., 2014). Anyone inside LAN (Local Area Network) can control most smart devices. Thus an adversary can capture a message with IP (Internet Protocol) information to act on behalf of the user. It could be classified as replay attacks when attackers reuse previous requests either towards the hub or the cloud-based service to gain information or control of the home (Geneiatakis et al., 2017). The adversary can steal a credential from a low-privileged user and try to escalate to gain unrestricted access to the SHS. Besides, security is an essential concern for the inhabitants, considering smart homes are intimate living environments, exposures could lead to physiological and financial issues (Heartfield et al., 2018).

Smart home solutions for access control that depend on an Internet connection have a central point of failure (Doan et al., 2018). Edge computing provides more security since data do not need to travel over a network to reach a server to be processed as in cloud computing (Hassan et al., 2018). It is inefficient to depend on data processing or storage outside the home. Since the devices and users are in the smart home, it should be self-sufficient. Attackers could also exploit the dependence on the cloud to perform Denial of Service (DoS) by isolating the house from the Internet. Cloud computing also facilitates eavesdropping and man-in-the-middle attacks due to the traffic exposure between the home and the server. Humayed et al. (2017) exposed one of the challenges for Cyber-Physical Systems (CPS) would be security by design, as isolating the LAN from the Internet. Besides, users lose control over their data by storing it in the cloud, facing privacy leakage risks (Khan et al., 2019).

Smart home users interact with devices using several interfaces (Zeng et al., 2017), like personal devices, home devices (e.g., smart assistants, tablets), or the devices themselves. Nowadays, many smart devices, such as smart switches or smart locks, can be controlled in the physical world. The main works on SHS security do not regard the multi-interface aspect by relying on the fact that users will always access the SHS using end-user devices, like smartphones or tablets. It can lead to security breaches once users would bypass the digital interface. Furthermore, the requirement of historical data in a security system contributes to breaches in the first days until it has collected and trained enough data to ensure its full functionality. It should be able to secure the smart home from the start and only consider the historical data to improve its accuracy naturally in the long term. Considering all the previous points, we investigate the main problem: *How to prevent and mitigate unauthorized access on the smart devices in SHS caused by stolen credentials to protect the security, safety, and privacy of the inhabitants?*

The research problem definition took into account an analysis of the state-of-the-art security works in smart homes, the comparison against the defined smart homes' security requirements, and the identification of open issues. The challenges to solve the problem are: to resist impersonation attacks; protect all access ways to the smart devices; work from day zero without historical data; and be independent of external networks and services. Therefore, we raised the following questions for this work to explore and answer:

1. *Which are the common approaches for access control in smart homes, and which of them could be the most effective to prevent and mitigate impersonation attacks?*
   The literature has revealed three main approaches: authorization, Intrusion Detection System (IDS), and continuous authentication. However, the last one proved the most appropriate to counter impersonation attacks once it continually verifies the users' authenticity. The Zero Trust (ZT) paradigm, which considers every interaction untrustworthy before verifying it, can also incorporate it. Furthermore, the more data sources involved in the access control decision, the more robust and precise it will be. Additionally, the user authenticity must be confirmed by strong features of the *Something You Are* class, which is harder to fake.

2. *How to protect all communication interactions and access ways between users and smart devices, including digital and physical means?*
   We have verified in the state-of-the-art that most works rely on a digital intermediary (smartphone, smart assistants, etc.) to enforce smart homes' security system. However, the users interact with the smart devices directly in the physical world. Therefore, the access control would not cover this interaction. Thus, it must be enforced on the final smart device and not rely on intermediaries to prevent the user from bypassing the security system.

3. *Among the approaches listed in the first question, what techniques can ensure operation from day zero?*
   In smart homes, few approaches currently depend only on instant data collected from context, devices, and users, for example, calculating with instant context trust or static device and user attributes. The ideal model must work from day zero with instant data and then, as time passes, improves with another module that counts on historical data.

4. *How can smart home access control be self-sufficient regarding data and processing independent of external resources?*
   Most smart home access control works rely on cloud computing or external resources. However, the security system should be independent of those resources for increased reliability. Furthermore, all sensors, data, and processing must be owned by inhabitants, with minimum collected data, to improve their privacy. Additionally, keeping the data traffic inside the local network contributes to less exposure and, consequently, more security.

5. *How robust, efficient, extensible, and performant can a Zero Trust continuous authentication for access control in smart homes against impersonation attacks be?*
   We must evaluate the access control robustness, efficiency, extensibility, and performance in a realistic smart home scenario with metrics testing whether the system meets all the requirements. The tests must consider changes in access control and environment to check for different use cases.

1.3 OBJECTIVES

This work aims to provide access control to avoid impersonation attacks while keeping the inhabitant's privacy with trustworthy continuous verification and a passwordless approach. Consequently, more people will feel safer adopting smart homes to improve their quality of life, and users will not depend on unreliable cloud service providers. They will be able to use their smart devices by any means. The Zero Trust (ZT) paradigm with the continuous authentication method guarantees that each request will be analyzed and user authenticity verified, supported by instant context information and user behavior, without requiring historical data. This work considers the following specific objectives:

- Survey and analysis of the state-of-the-art security works in smart homes especially focused on countering impersonation attacks;
- Definition of requirements for a security system for smart homes;
- Proposition of a smart home access control model to resist impersonation attacks, to fill the gaps of the state-of-the-art, and that satisfies the defined smart homes security requirements;
- Investigation of the robustness, efficiency, extensibility, and performance of applying a multilayer approach by enforcing strict rules, verifying the environment's trust, and monitoring user behavior.

1.4 CONTRIBUTIONS

The development of this dissertation resulted in scientific contributions in the area of computing, emphasizing access control for smart home networks with a client-server architecture. The contributions of this work are detailed below:

- Survey of the main threats for smart home networks and the state-of-the-art works containing countermeasures to these threats. Three main approaches were deeply studied: authorization, Intrusion Detection System (IDS), and continuous authentication. The discussion of the main works revealed their strengths and limitations, as well as the requirements for a smart home network focusing on user privacy and security against impersonation attacks.
- Proposal of ZASH (Zero-Aware Smart Home), a system to provide access control for an SHS using continuous authentication with Zero Trust (ZT) to limit the amount of required data, powered by edge computing to dismiss unreliable service providers and capable of processing requests originated from any means. We designed the system with user levels (e.g., admin, adult, child, visitor), device classes (e.g., critical, non-critical), and actions (e.g., view, control, manage) to mitigate impersonation attacks by contributing to device isolation and user action differentiation. The continuous authentication process counts on three phases. The first verifies users, devices, and actions using ontology rules. The second phase verifies context information to check if they achieve the expected trust for a requested action with a specific user level on a device class. The final step verifies whether the requested action probability exceeds a threshold considering a Markov Chain built on all previous activities.
- Evaluation of the robustness, efficiency, extensibility, and performance of ZASH to protect users' security, safety, and privacy against impersonation attacks. We analyzed the proposal in an evaluation in Python to verify the model's logic by collecting general metrics and exploring individual use cases. Then we examined ZASH in an assessment in the ns-3 network simulator by collecting metrics to validate all the requirements for an SHS. The metrics also be considered contributions since they refer directly to one of the requirements. The results revealed that ZASH respects users' privacy, has a fast response time, copes with adding and removing devices, blocks most attacks (99% with a proper configuration), isolates smart devices, and enforces access control for all types of interactions.

1.5 TEXT STRUCTURE

We organized the dissertation into six chapters. Chapter 2 describes the necessary foundations for a complete understanding of the work. It starts with an overview of the IoT, passing through CPSS, IoE, and the application of smart homes. Then, it clarifies the characteristics and requirements for security in IoT, detailing user privacy, access control, continuous authentication, Intrusion Detection Systems, and impersonation attacks. It also broaches decision reasoning with context-aware, ontology modeling, behavior-based, and Zero Trust approaches. Lastly, it describes how the decision can be computed with cloud and edge resources. Chapter 3 presents the main threats and countermeasures in smart homes, then the related work as the state-of-the-art on continuous authentication, authorization, and Intrusion Detection Systems, followed by a discussion with general relation of the works with the security requirements for smart homes. Chapter 4 introduces the proposed solution, ZASH, which employs access control with continuous authentication based on context and behavior decisions to secure smart homes against impersonation attacks using edge computing and Zero Trust. Chapter 5 details the evaluations to validate ZASH's robustness, efficiency, extensibility, and performance. Finally, Chapter 6 presents the final considerations about the research until the conclusion of this work, as well as possible future works.

2 BACKGROUND

This chapter presents the necessary foundations for the understanding of this research work. Section 2.1 broaches the IoT paradigm, including its evolutions towards CPSS, IoE, and the smart home application. Section 2.2 discourse about security in the IoT, including user privacy, access control, continuous authentication, Intrusion Detection System (IDS), and impersonation attacks. Section 2.3 presents the decision reasoning process, comprising the context-aware, ontology modeling, behavior-based, and Zero Trust (ZT) approaches. Finally, Section 2.4 introduces the decision computing process with cloud and edge resources.

2.1 INTERNET OF THINGS

The Internet comprises many networks that use standard protocols and provide joint services. Initially, it was planned and controlled by a central entity, but its expansion took place decentralized (Tanenbaum et al., 2011). The origins of this network occurred in the context of the Cold War in late 1969 with the ARPANET due to its link with the ARPA (Advanced Research Projects Agency), a centralized organization for defense research. ARPANET was created to correct a flaw in the telephone system at the time, which depended on exchanges for communication between cities. This vulnerability could be exploited in an eventual attack. Thereby, the idea was to create a distributed switching system. The first ones were among the four research centers with outstanding performance in ARPA in the United States: the University of California at Los Angeles (UCLA), the University of California at Santa Barbara (UCSB), Stanford Research Institute (SRI), and the University of Utah (UTAH).

ARPANET's expansion was rapid; in September 1972, the network already had 34 nodes. These connections allowed sharing of information among researchers, which helped a lot in developing research. Soon companies and other institutions not linked to ARPA became interested in this communication technology, and the NSF (National Science Foundation) created the NSFNET. After its success and enormous expansion, NSF stimulated the creation of ANS (Advanced Networks and Services), a non-profit company formed by MERIT, MCI, and IBM. After that, the government left the networking business by hiring four operators to establish network access points, or NAPs (Network Access Points), ensuring that all regional networks could communicate. Networking technologies developed rapidly, and soon Europe and elsewhere created similar infrastructures. Internet use exploded with the rise of personal computers and the emergence of the World Wide Web (WWW) in the early 1990s. The Internet first came to people with DSL (Digital Subscriber Line), which reuses the infrastructure of telephone lines. Data transmission speed and capacity increased with broadband and with the use of optical fiber. Currently, there is equipment and infrastructure to support gigabit (up to 940 Mbps), gig+ (up to 2 Gbps), and multi-gig (up to 10 Gbps) traffic for many users in the world (Domingo, 2023).

The already established Internet of Things (IoT) paradigm comes from ubiquitous computing, the idea of computational devices so pervasive in people's daily lives that their presence goes unnoticed (Weiser, 1991). Zorzi et al. (2010) and Ortiz et al. (2014) employed the term Intranet of Things to define a local network with a set of objects, such as Wireless Sensor Networks (WSNs), Machine-to-Machine (M2M) communication, and smart homes. In this context, these networks work in isolation and extract only local information with specific content about the objects. With the rise of the Internet, once-isolated networks found a way to interact. The IoT can deliver large-scale, comprehensive, historical information through this collaboration across the Intranets of Things, overcoming the heterogeneity of devices, communication technologies, and deployment goals. In addition to the possibility of integrating existing systems, the IoT favors the emergence of new applications and services dedicated to different purposes. The IoT is a paradigm in which various objects communicate with each other and cooperate to achieve a common goal (Atzori et al., 2010).

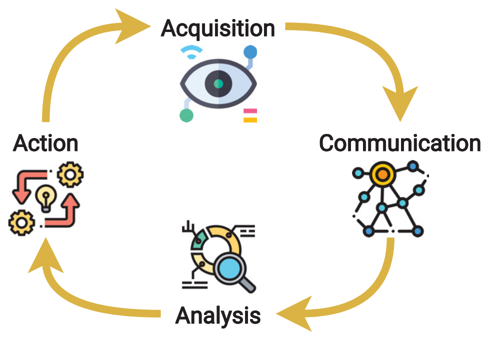


Figure 2.1: IoT life cycle

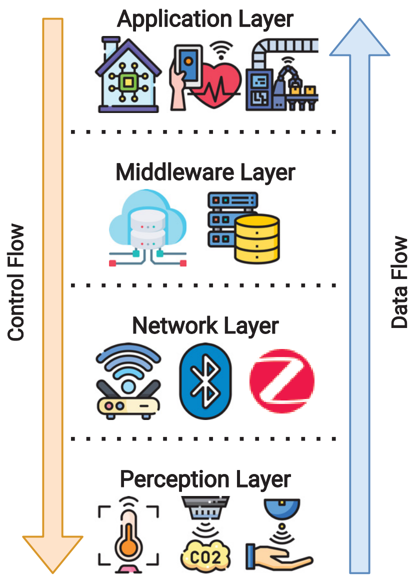


Figure 2.2: IoT architecture

According to Rahman et al. (2018), the IoT life cycle is composed of four phases, as shown in Figure 2.1: acquisition, when the sensors collect data from the physical environment; communication, when the system sends the collected data in the previous phase through the network to the destination device or other data centers; analysis, when data is processed and analyzed to provide useful information; and action, when the info proceeds to further actions on the physical or digital environments. Figure 2.2 details a four-layer IoT architecture, as exposed in Chen et al. (2018); Al-Naji and Zagrouba (2020), composed by: perception layer, where sensors collect data and actuators interact with the physical world, enabled by technologies like RFID (Radio-Frequency Identification) tags, cameras, smart lights, smart locks, etc.; network layer, responsible for connecting smart things to devices and servers, transmitting data from the physical layer to the middleware layer through wired or wireless networks (e.g., Bluetooth, ZigBee, Wi-Fi (Wireless Fidelity), 5G, NFC (Near Field Communication)); middleware layer, that stores, analyses, and process vast amounts of data by employing database, cloud computing, and big data; application layer, which represents the industry logic, being the frontend of IoT architecture and in charge of delivering specific services to the user.

The main challenges to implement an IoT network from Butun et al. (2020) are: heterogeneity, once it's composed by a diverse range of devices, including gateways, switches, sensors, actuators, smart appliances, mobile systems, etc.; scalability, because IoT usually requires vast amount of devices that need addressing, naming, managing and servicing; communications, since various wired and wireless technologies are employed depending on requirements like coverage and energy constraints, such as Bluetooth, Zigbee, and LPWAN (Low Power Wide Area Network); energy consumption, being one of the main challenges, hence programs need to have lightweight processing; data privacy, once devices can reveal sensitive information about people, such as location and health condition; self-awareness, as smart objects should be autonomous and adapt to the environment in real-time and without human intervention; and interoperability, because of the heterogeneity characteristic of devices to communicate, collaborate and share data, this process should follow standards.

Application domains are diverse when it comes to IoT due to their potential. Objects in all environments surround people, but only a few are connected to a network and transmit information. IoT can improve people's quality of life, whether at home, traveling, undergoing medical treatment, at work, or at the gym, among other situations. According to Atzori et al. (2010), several areas can benefit from using IoT, such as:

Transport and logistics: in transport, there are advances in technology in cars, trains, and bicycles, as well as highways and rails, which have been equipped with sensors, actuators, and energy processing. Identifiers and sensors have helped to manage traffic in cities, as well as track products in transport. Using technologies such as RFID and NFC allows information processing in real-time, enabling better traceability of consumer goods in the production and distribution chains. Autonomous cars and augmented reality maps to assist tourists are also possible applications in this area.

Healthcare: several benefits emerged from IoT in healthcare. Among them is real-time tracking of people and objects in motion, for example, to monitor the flow within a hospital to optimize processes and control inventories. It allows the identification and authentication of people for records in hospitals to maintain a history of care and prevent the use of inappropriate medications. It enables automatic data collection and analysis with Machine Learning for automated appointments, audit procedures, and medical inventory management. The use of IoT through sensors helps in the real-time monitoring of patients to support diagnoses and health indicators. Finally, integrating different wireless network technologies helps in the resilience of biological signals monitoring in patient mobility situations.

Smart environments (smart homes, smart cities): in this context, sensors and actuators distributed in homes or offices help to provide greater comfort for people. They enable regulating the ambient temperature, lights, power, and electronic equipment control from a distance. Industry 4.0 also relies on these technologies to ensure rigorous quality control and production planning based on statistical data collected from factory sensors. In leisure, museums and smart academies are existing examples. In addition, smart cities are gaining more and more encouragement, such as Curitiba, which through *Vale do Pinhão* and other initiatives, foster an ecosystem of innovation (Gazeta do Povo, 2019).

Personal and social domain: applications of this domain enable interaction between people to maintain or build social relationships. Practical examples include using RFID and Wi-Fi to generate events that people can share on social networks like Facebook or Twitter. Furthermore, analyzing historical data to identify trends in activity over time emerges as another application. Search engines for lost objects through RFID, providing the last positions or the current location, refer to another example in this area. Finally, burglary alerts when detecting if an object has left a particular restricted area using sensors is another possibility.

Futuristic applications: applications that do not yet have the necessary technologies for their implementation or that society is not yet prepared to use (Atzori et al., 2010) fit into this domain. It is possible to mention autonomous taxis, which would optimize users' waiting time and commute by using real-time traffic data and communicate with each other to distribute well in the regions. Another application would be an Information City Model (ICM), which would provide information about the city's buildings and infrastructure, such as sewage, power grid, train lines, bus corridors, etc. In this way, buildings could share energy and other resources to optimize cost-effectiveness and supply and demand. Planning and projects would be done intelligently based on statistics. Lastly, an improved game room equipped with sensors for location, motion, acceleration, humidity, temperature, noise, voice, visual information, heart rate, and blood pressure emerges using data collected from these sensors to provide an immersive experience in computer games. Interactive scenarios with touch sensors are also an application possibility using IoT.

2.1.1 Cyber-Physical Social Systems (CPSS)

Cyber-Physical Systems (CPS) consist of services with computing capabilities, like storage, processing, and inference, and physical devices present in the physical world to gather information and act upon the environment (Humayed et al., 2017). Embedded computers and communication networks govern physical actuators that operate outside and receive sensor inputs, creating a smart control loop capable of adaptation, autonomy, and improved efficiency (Zanero, 2017). The intersection of applications of CPS with IoT is normally big. Nevertheless, while IoT relates to the technology that enables the interconnection of all types of devices through the internet to exchange data, optimize actuators, and monitor devices to generate results, the CPS consists of computation and control components tightly combined with physical processes, providing the foundation for IoT and bringing about advanced efficiency and connectivity of devices, systems, and services in countless domains (Vanderbilt, 2020). According to Humayed et al. (2017), examples of CPS applications are:

Industrial Control Systems (ICS): sometimes ICS is called SCADA or distributed control systems. It comprises different controllers with different capabilities to enhance the control, monitoring, and production in different industries, such as nuclear plants, water and sewage systems, and irrigation systems. It uses sensors and actuators to gather information, computers to process, and wireless or wired networks to control the operation. The Industrial Internet of Things (IIoT) shares the same characteristics and objectives.

Smart grid systems: the next generation of the power grid provides enhanced control for the governments over the energy distribution, whereas it allows home consumers to monitor their usage, which would be beneficial economically and environmentally. The smart grid depends on power application, where the core functions of the smart grid are provided, i.e., electricity generation, transmission, and distribution. It also relies on supporting infrastructure, the intelligent component comprising software, hardware, and communication networks.

Medical devices: it consists of physical devices to deliver health care services, such as IMDs (Implantable Medical Devices) or wearable devices. They collect patient information and communicate through wireless networks with another device with more processing power and storage. The information provided by these devices helps in monitoring and medical decisions.

Smart cars: also called intelligent vehicles, they are environment-friendly, fuel-efficient, safe, and equipped with enhanced entertainment and convenience features. These cars rely on several Electronic Control Units (ECUs) responsible for monitoring and controlling various functions, like engine emission control, brake control, multimedia players, and cruise control.

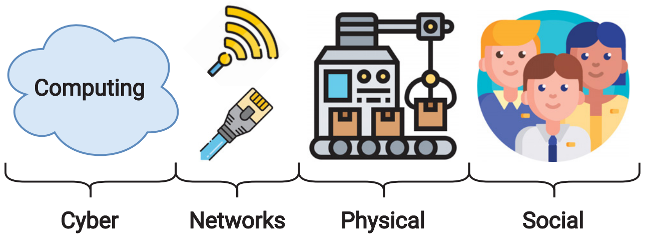


Figure 2.3: Cyber-Physical Social Systems

The Cyber-Physical Social Systems (CPSS) incorporate human factors with CPS to embrace human knowledge, mental capacity, and sociocultural elements (Dong and Ansari, 2020), as shown in Figure 2.3. This new paradigm results from technological development, including cloud computing, smart grids, autonomous automotive systems, medical monitoring, process control systems, distributed robotics, and mobile networks. It provides an enhanced user experience regarding energy efficiency, reliability, security, and cost-efficiency with human factors. The challenges of CPSS are integrating human activities into computing, resource management, and scheduling in computing infrastructures to optimize reliability and scalability and coordinate various services in heterogeneous platforms.

2.1.2 Internet of Everything (IoE)

The Internet of Everything (IoE) refers to the network of connections between smart things, people, processes, and data with real-time data/information flows between them (Langley et al., 2021). IoE comprises other connection-based paradigms such as IoT, Internet of People (IoP), and IIoT. There are endless opportunities for improving our daily activities when everything is connected through billions and trillions of connections. Being aware of everything in the environment enables faster response times to medical or public safety emergencies and saves lives, improves the quality of citizen life by providing direct and personal services from the government, and discovers new information about how our cities work, thus enabling city leaders to use resources more efficiently and save money while providing superior services (de Matos et al., 2017). As detailed in Figure 2.4, IoE involves not only M2M communication but also People-to-Machine (P2M) and People-to-People (P2P) communication through technology (Shinkarenko, 2020).

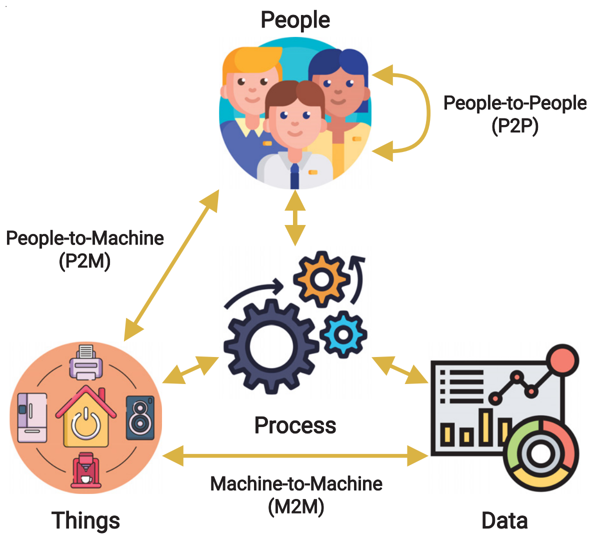


Figure 2.4: Internet of Everything

According to de Matos et al. (2017); Shinkarenko (2020) and as detailed in Figure 2.4, the foundations of IoE are: things, which refer to connected objects that gather information about their status through sensors and share this information with one another over the Internet, being expanded to any physical object in IoE; people, which not only play an essential role in P2M and P2P communication, but also are the core of IoE, since humans use connected devices every day, analyze data, and harness the potential of data insights; data, crucial for the decision making process and is increasingly being transformed in more complex by the things themselves due to edge computing, thus improving network control, storage for latency critical applications, and security improvement for the processing of data at edge devices (Ning et al., 2019); and process, determining how each of the elements above works with the rest to provide more excellent value in the digital world, assuring the correct information is delivered to the right person at the right time in the appropriate way. As explained in Shinkarenko (2020), IoT consists of connected things, as exposed in Section 2.1.2, and IoE expands this paradigm to include connected things, people, and data in a unified process Section 2.1.2.

$$IoT = network + things \tag{2.1}$$

$$IoE = network + things + people + data + process \tag{2.2}$$

The main challenges of IoE from de Matos et al. (2017) are: automated configuration of data providers, due to the billions of data providers to be connected together over the Internet; context discovery, to provide meaning to the vast amount of produced data; acquisition, modelling, reasoning, and distribution, as it is essential to define and follow a standard specification so different techniques can be added to the solutions without significant effort; selection of data providers, since there will be several data providers to the same environment; security, privacy, and trust, once data is collected in large scale, it needs to be protected in all layers of sensor hardware, communication protocol, context annotation and context discovery, context modelling, and the context distribution; scalability, as the data traffic grows, it requires sufficient bandwidth that might be enabled by recent technologies of 5G and, in the future, 6G (Padhi and Charrua-Santos, 2021); reliability, as critical applications, like healthcare, safety and security applications, utility functions, and industrial systems, demand for continuous, uninterrupted, real-time communication; context sharing and interoperability, mostly neglected as new systems emerge with new concepts and architectures. Specifically, fog computing in IoE presents challenges related to the user's privacy, resource allocation, and unavailability of testing software and programming models (Ning et al., 2019).

2.1.3 Smart Homes

Smart homes are a ubiquitous branch of computing dedicated to embedding intelligence in homes for comfort, healthcare, safety, and energy conservation (Alam et al., 2012). Several terms refer to smart homes, such as smart house, home automation, domotique, intelligent home, adaptive home, and aware house. We can categorize the projects in this area in: comfort, such as activity identification and event automation, where context awareness relates to an important prerequisite, and remote access control, which usually provides bidirectional communication between the home and the user through the Internet; healthcare, with local monitoring, which helps to identify health conditions, ensure assistive services, and generate local warnings or alarms, and remote monitoring, that delivers medical information to caregivers and doctors; and security, with user authentication and device authentication. Further, *smart homes* can be defined as focused on the home and user as a highly automated residential building with integrated appliances, emphasizing modern technology, convenience, and (domestic) efficiency. It can also focus on building and systems such as providing energy performance, ancillary services, and distributed energy generation and how to address them using information and communication technology (Mocrii et al., 2018; Darby, 2018). Both perspectives share the reliance on the network to connect devices to enable remote access and control and to provide services.

According to Edwards and Grinter (2001); Dey (2001); Mocrii et al. (2018), the intelligence in smart homes and ubiquitous computing has the following characteristics: the environment can use sensor data to know states in the physical world; the environment can infer activities from a set of data; the environment may predict intentions and interests based on current states and activities; and the system may preemptively act on assumptions of intent. Bakar et al. (2016) exposed the smart term of smart homes refers to the inhabitants' capacity to monitor their activities and provide the necessary support to assist daily activities, even for security or safety purposes. The sensors of a smart home are responsible for collecting data from the environment, transformed into information, and analyzed in different applications. They are spread in strategic places throughout the house and are identified uniquely by IDs that, combined with timestamps, generate recorded events. The sensors can be equipped with wired and wireless communication technologies, with ZigBee and Wi-Fi as the main ones (Bakar et al., 2016). We can classify them into two groups related to user privacy: the obtrusive type dense sensing (Poppe, 2010), which poses more risks to user's privacy, such as body-worn sensors, cameras, and microphones; and the none-obtrusive type dense sensing (Chen et al., 2012), less risky to user's privacy, such as PIR (Passive Infrared) sensors, motion detectors, door/window entry point sensors, electric usage power sensors, and pressure sensors. The non-obtrusive class is cheap, reliable, readily available, and could be deployed in large quantities (Choudhury et al., 2006; Bakar et al., 2016).

As presented by Mocrii et al. (2018) and shown in Figure 2.5, a general cloud-based smart home architecture is composed of an internal network with end devices, sensors, appliances, and actuators. The gateway is located at the network's edge and is responsible for communicating with the external Internet. It generally supports multiple communication protocols for interoperability with the end devices with enough processing power to prepare data before sending it to the cloud. The gateway should also filter outside commands with a security layer to protect the internal network. In turn, the cloud can integrate with other third-party services, such as data visualization, smart home device management, or user access and role management (Mocrii et al., 2018). Other works proposed architectures with similar structures for smart homes, such as Soliman et al. (2013); Jie et al. (2013); Zhou et al. (2013).

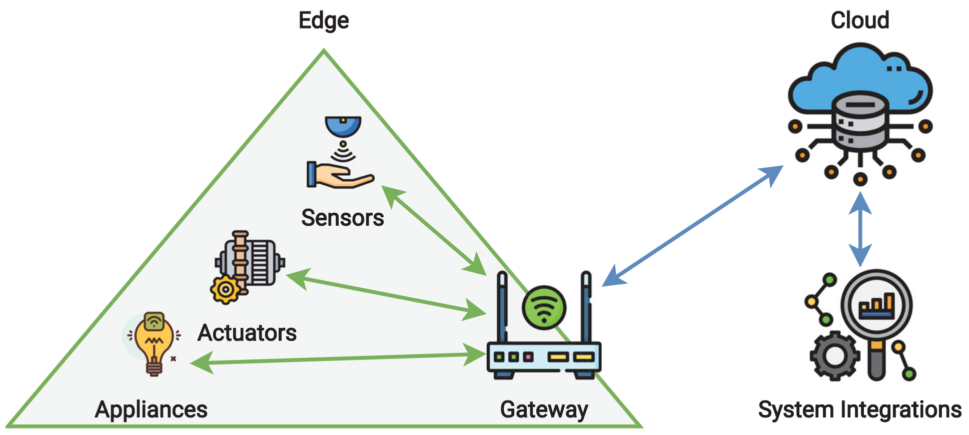


Figure 2.5: Cloud-based smart home architecture

The operating system requirements for IoT devices from Mocrii et al. (2018); Milinković et al. (2015); Devopedia (2020) contain: small footprint due to the resource constraint factor in the end devices; scalability, as it needs to scale considering available resources and the CPU architecture; modularity, since devices differ in their needs depending on their functions; portability, to run on different types of hardware and architectures; connectivity, since devices need to communicate with other ones; security, as it should be able to add security measures as needed; reliability, as devices should be able to run for long periods without errors; and run programs concurrently so that it can fulfill the tight deadlines in a real-time application. Some popular IoT Operational Systems (OSs) are Contiki, TinyOS, RIOT, Mantis, FreeRTOS, LiteOS, Android Things, etc. According to Mocrii et al. (2018), the data acquisition in smart homes can come from three sources: active user interaction directly generated by the user, such as voice commands, gesture recognition, triggering actions from pushing a button and interacting with a touch screen; passive user interaction generated as a result of human activity, such as motion detector sensors, video from cameras, silhouettes from depth cameras, RFID tags, smart floor sensors, and wearable body sensors; non-user data generated by devices, such as thermostat readings, humidity sensors, and airflow sensors.

The connectivity of smart devices in smart homes can depend on many wired or wireless communication protocols. Body Area Networks (BAN), Personal Area Networks (PAN), and LAN are commonly applied to form Home Area Networks (HAN) in the context of smart homes. Zheng et al. (2014) noted that the Wireless Personal Area Networks (WPAN) or Wireless Local Area Networks (WLAN) features to measure Quality of Service (QoS) are latency, transmission power, reliability, and bandwidth. While wired communication protocols provide security, ease of use, distance, data rate, and reliability, they also have higher costs and complexity, low mobility, require power, and are hard to expand. Smart homes' most commonly employed wired technologies are Ethernet, X10, UPB, INSTEON, MoCA, and KNX. Alternatively, wireless communication protocols yield mobility, expandability, lower costs, and flexibility, but they also bring insufficient security and data rates, susceptibility to interference, and complex coverage. Smart homes' most popular wireless technologies are Wi-Fi, Bluetooth, ZigBee, Z-Wave, and 6LowPAN. Mocrii et al. (2018) identified four core elements of smart home technologies: technical and social disruptiveness; the need for adaptation and familiarization from householders; the difficulty of and little support for learning to use; and the lack of evidence of substantial energy savings and risk of energy intensification.

The Connectivity Standards Alliance (CSA), formerly the Zigbee Alliance, refers to a group of companies, including Amazon, Apple, Google, and Samsung, that develop, certify, and promote IoT technology standards through a well-established, collaborative process (CSA, 2023a). Recently, CSA (2023c) proposed the Matter protocol as an industry-unifying standard to promise reliable, secure connectivity. It is a seal of approval that devices will always work seamlessly together. Matter creates more connections between more objects, simplifies development for manufacturers, and increases consumer compatibility. It is built with market-proven technologies using Internet Protocol (IP) and is compatible with Thread and Wi-Fi network transports. The Matter development goals from CSA (2023b) are to be unifying, built with and on top of market-tested, existing technologies; interoperable, permitting communication between any Matter-certified device, subject to users' permission; secure, leveraging modern security practices and protocols; user control, end-user controls authorization for interaction with devices; federated, with no single entity serving as a throttle or a single point of failure for the root of trust; robust, as the set of protocols specifies a complete lifecycle of a device, starting with the seamless out-of-box experience, through operational protocols, to device and system management specifications required for proper function in the presence of change; low overhead, practically implementable on low compute-resource devices, such as MCUs; pervasive, broadly deployable and accessible, by leveraging IP and being implementable on low-capability devices; ecosystem-flexible, flexible enough to accommodate deployment in ecosystems with differing policies; easy to use, providing smooth, cohesive, integrated provisioning and out-of-box experience; open, as the project's design and technical processes are open and transparent to the general public, including non-members wherever possible.

As specified by CSA (2023b), the Matter interactions flow through the following stack:

1. Application: High-order business logic of a device. For example, an application focused on lighting might contain logic to handle turning on/off the bulb and its color characteristics.
2. Data Model: The data layer corresponds to the data and verb elements that help support the application's functionality. The application operates on these data structures when there is an intent to interact with the device.
3. Interaction Model: The Interaction Model layer defines a set of interactions that can be performed between a client and server device. For example, reading or writing attributes on a server device would correspond to application behavior on the device. These interactions operate on the elements defined at the data model layer.
4. Action Framing: Once an action is constructed using the Interaction Model, it is serialized into a prescribed packed binary format to encode for network transmission.
5. Security: An encoded action frame is then sent down to the Security Layer to encrypt and sign the payload to ensure that data is secured and authenticated by both sender and receiver of a packet.
6. Message Framing & Routing: With an interaction encrypted and signed, the Message Layer constructs the payload format with required and optional header fields, specifying the message's properties and routing information.
7. IP Framing & Transport Management: After the final payload has been constructed, it is sent to the underlying transport protocol for IP data management.

2.2 SECURITY IN IOT

With most systems already digitized, the connected world requires secure solutions to maintain people's privacy and the correct service functionality. The three essential components of computer science security from Butun et al. (2014) are: prevention, which aims at preventing attacks before they happen, as intrusion prevention mechanisms that resist external attackers but are not usually designed to withstand internal attackers; detection, which is responsible for detecting compromised nodes caused by a successful attack, as using an Intrusion Detection System (IDS) that identify the attack and triggers the mitigation mechanism; mitigation, which acts to reduce damage caused by an attack, as dismissing the affected nodes in a network or disabling the ports of a computer exploited during the attack. These components cannot be considered separately in the defense system as they complement each other in the security structure.

The IoT emergence due to the development of the WSNs comes with challenges for the security of its services beyond those mentioned in Section 2.1. As stated by Khattak et al. (2019), the security requirements of IoT are: confidentiality and privacy, as only authorized access to information must be allowed and unauthorized access prevented; integrity, as modification in the data, must be controlled to avoid the addition of fake information, replication of old data, data steal or deletion; availability, to guarantee the services and data will be available when needed by authorized users; secure communication, due to the open and broadcast nature of IoT, it is vital to prevent attacks, such as spoofing and eavesdropping; access control, which deals with identity and access management; authentication, that ensure the users are whom they claim to be; non-repudiation, to prevent entities from denying their actions.

As exposed by Humayed et al. (2017), the CPS presents unique security breaches and threats related to five factors: source, target, motive, vector, and consequence. The attacker (source) exploits a target with a reason to generate outcomes. The source can be adversarial when it has malicious intentions, accidental when caused accidentally by legitimate entities, or environmental when including natural disasters (e.g., floods, earthquakes), human-caused disasters (e.g., fires, explosions), and infrastructure failures (e.g., power outage, telecommunications loss). Targets are applications and their components or users (Cebula and Young, 2010; Kang et al., 2009; Nicholson et al., 2012; Ross, 2012; Stoneburner et al., 2002; Stouffer et al., 2011; US-CERT, 2009). Motives are the reasons to launch an attack, such as criminal, spying, terroristic, political, or cyberwar (Setola, 2011; US-CERT, 2009). Vectors are the mechanisms practiced by the attack, such as interception, interruption, modification, or fabrication (Pfleeger and Pfleeger, 2016). Consequences can compromise CPS confidentiality, integrity, availability, privacy, or safety. The main threats of a CPS are: criminal, financial, political, privacy, and physical.

The attacks in IoT can be classified as passive or active (Butun et al., 2020). In passive, the attacks compromise mainly confidentiality as they cannot be noticed during their execution because the attackers are camouflaged, i.e., hidden, and can be grouped in eavesdropping, node malfunctioning, node tampering/destruction, node outage, and traffic analysis types. The active attacks usually compromise the confidentiality and integrity of data, try to obtain unauthorized access and disturb communication. We can group them according to the IoT layers: physical, MAC (Media Access Control), network, transport, and application. In the physical layer, the attacks can be Denial of Service (DoS) caused by jamming and node tampering. The MAC layer attacks comprehend collusion, denial of sleep, de-synchronization, exhaustion, link layer flooding, link layer jamming, spoofing, unfairness, and the 6LoWPAN exploit. The network layer contains the following attacks: HELLO-flooding, hole attacks (blackhole, grayhole, sinkhole, wormhole), node replication, routing attacks, RPL exploit, and sybil. The attacks from the transport layer can be de-synchronization, MQTT exploit, session hijacking, and SYN-flooding. The application layer attacks comprise CoAP exploit, false data injection, path-based DoS, re-programming, and sensor overwhelming.

The following subsections detail other aspects of security important for comprehending this work: user privacy, access control, continuous authentication, Intrusion Detection Systems (IDS), and impersonation attacks.

2.2.1 User Privacy

The concept of privacy is the right of individuals to maintain confidentiality and control over their information (Porambage et al., 2016). Deploying a vast amount of sensors and devices generates enormous amounts of data in the context of IoT. This data contains information that could compromise people's privacy. An attacker can appropriate this data to track, locate, profile, and even blackmail its owners. Furthermore, applying techniques on large amounts of information reveals trends and behaviors that can be exploited (Porambage et al., 2016). This problem becomes even more oriented toward people in IoE, where personal data is generated, transported, processed, and possibly stored. Organizations and governments are defining regulations and concepts to address these issues.

According to the General Data Protection Regulation (GDPR) (Intersoft Consulting, 2019), personal data carries information about a person, such as direct identification, like first name, last name, and telephone number. In addition, this category includes data employing pseudonyms or non-directly identifying information that does not allow the direct identification of users but allows for the individualization of behaviors. GDPR encourages utilizing non-directly identifiable data to minimize the risks of an eventual leak. GDPR refers to sensitive data as those that need special protection because of their nature or relationship to individuals' fundamental rights and freedoms. The European regulation, which served as the basis for the Brazilian General Data Protection Law (LGPD, *Lei Geral de Proteção de Dados* in Portuguese), considers sensitive data those referring to ethnic or racial origin, political opinion, philosophical or religious beliefs, group affiliations, genetic data, biometric data in order to identify individuals and related data health, sexual life or sexual orientation. In particular, this work counts on personal data generated in the people's intimacy inside their homes.

GDPR prohibits the processing of sensitive data unless the party involved has given its explicit consent within the scope of legitimate activities carried out by associations or foundations whose objective is to allow the exercise of fundamental freedoms if there is a public interest based on the current legislation of all countries of the European Union, for example, in the working environment, social protection, pensions, health and other severe threats to health. This work considers data sharing with the explicit consent of its owners. Law 13.709, known as LGPD, is the Brazilian version of the GDPR. The LGPD was sanctioned by Michel Temer in August 2018 and wholly entered into force in August 2021. It applies to any activity involving the use of personal data, including in digital media, by a natural or legal person governed by public or private law to protect the fundamental rights of freedom and privacy. Figure 2.6 exposes its main goals. The law also applies extraterritorially if the data processing operation is carried out in the national territory; the processing activity has as its objective the offer or supply of goods or services or the processing of data of individuals located in the national territory; the personal data, the object of the processing, have been collected in the national territory (LBCA, 2021).

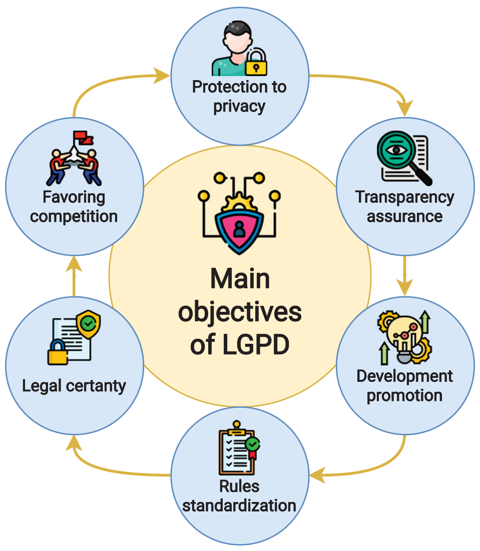


Figure 2.6: Main objectives of LGPD (LBCA, 2021)

Data holders have the right, guaranteed by article 18, concerning their personal data: confirmation of the existence of processing; access; correction; anonymization, blocking or deletion; portability; obtaining information about sharing; the revocation of consent (LBCA, 2021). This law should affect the development of new research and applications in IoT since they generate a large amount of data collected from sensors, which may be related to human activities or behavior. The applicability of these restrictions of the new law will depend on the inspection and subsequent understandings defined by the courts.

In the context of smart homes, the breach of physical privacy is as worrisome as data privacy. For instance, an attacker can monitor the house transmissions to know the presence of inhabitants or their lifestyles (Heartfield et al., 2018; Ghansah, 2009). Many attacks target the cameras and microphones in houses as they are usually poorly secured. The baby-monitor devices are also vulnerable, and many reports noticed breaches on them, which permit attackers to spy on children visually (Albrecht and Mcintyre, 2015; BBC, 2020; Heartfield et al., 2018). According to Moreham (2014), the breach of physical privacy means being watched, listened to, or recorded against one's wishes. User privacy violations in smart homes often affect the inhabitant's behavior, causing shame, inconvenience, impotence, and loss of control (Heartfield et al., 2018).

2.2.2 Access Control

Access control limits what legitimate users can do in a computer system. Beyond limiting users' actions and operations, it also restricts what programs they execute can do. The main goal of access control is to prevent activity that could lead to a security breach (Sandhu and Samarati, 1994). Access is the data flow between a subject (e.g., person, process, program) and an object (i.e., a system resource, file, printer). It allows only authorized users access to appropriate data and denies access to unauthorized users. Access control contains security mechanisms that control how subjects can interact with objects (Khattak et al., 2019). As detailed in Figure 2.7, the components of access control are: identification, which is unique per user and identifies them univocally and publicly (e.g., user ID, account number, RFID, IP, MAC address); authentication, which proves identity is legit; authorization, what rights and permissions users have; auditing or accounting, which keeps the non-repudiation property by linking actions to subjects. Figure 2.8 represents the main models for implementing access control, and the following paragraphs describe them.

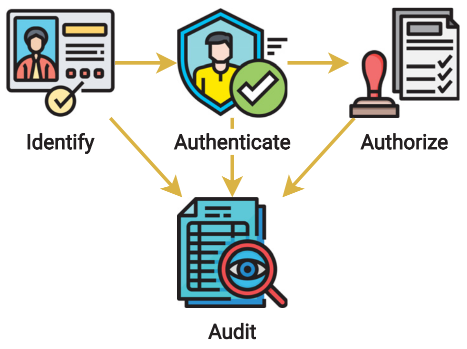


Figure 2.7: Access control components (IAAA)

Discretionary policies: they specify what are the allowed access modes (e.g., read, write, or execute) of users (or groups of users) on the objects, being very flexible but less secure. The Discretionary Access Control (DAC) can be implemented using an authorization table (popular in DBMS - Database Management Systems), access control matrix (not scalable), or Access Control Lists (ACLs, popular in OSs) (Sandhu and Samarati, 1994). Although ACL is popular, it is centralized by nature, cannot support different levels of granularity, is not scalable, and is prone to a single point of failure (Ouaddah et al., 2017). The Capability-Based Access Control (CapBAC) relies on granting rights to an entity possessing the capability and can be applied through Capabilities Lists (CLs) or by using a token, ticket, or key (Ouaddah et al., 2017; Dennis and Van Horn, 1966).

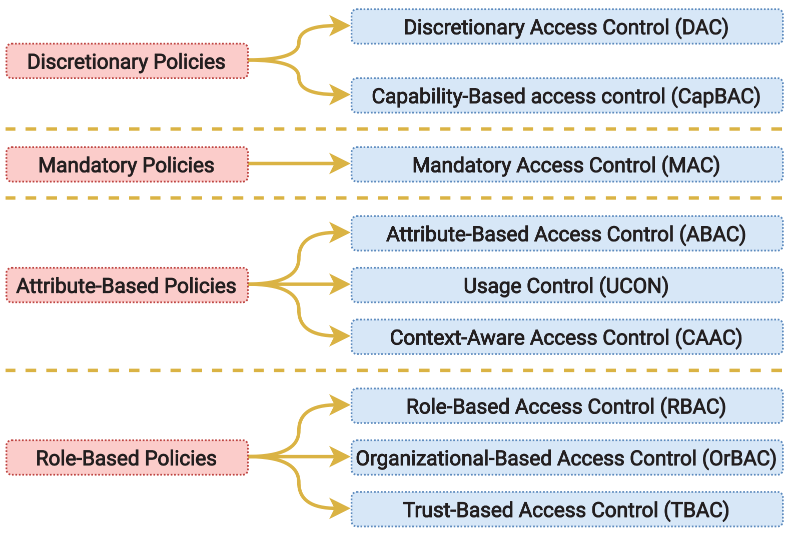

Discretionary Policies
Discretionary Access Control (DAC)
Capability-Based access control (CapBAC)
Mandatory Policies
Mandatory Access Control (MAC)
Attribute-Based Policies
Attribute-Based Access Control (ABAC)
Usage Control (UCON)
Context-Aware Access Control (CAAC)
Role-Based Policies
Role-Based Access Control (RBAC)
Organizational-Based Access Control (OrBAC)
Trust-Based Access Control (TBAC)

Figure 2.8: Access control models

Mandatory policies: they define hierarchically organized security levels associated with the users (trustworthiness) and to the objects (sensitivity), for example, from the multilayer implementation: Top Secret (TS), Secret (S), Confidential (C), and Unclassified (U), where TS > S > C > U. The Mandatory Access Control (MAC) can be implemented to protect confidentiality with the Bell-LaPadula model (Bell, 1973), where the properties are No-Read-Up (subjects cannot read objects of higher levels) and No-Write-Down (subjects cannot write on objects of lower levels, preventing information leak). It can also protect information integrity with the Biba model (Biba, 1977), where the properties are No-Write-Up (subjects cannot write on objects of higher levels), and No-Read-Down (subjects cannot read objects of lower levels because they are unreliable). We can also implement MAC with categories that do not have hierarchical order.

Attribute-based policies: they incorporate the Attribute-Based Access Control (ABAC), where the access is defined by evaluating attributes of the subject, object, requested operations, and environment conditions against policies, rules, or relationships describing which operations are allowed for the set of attributes (Hu et al., 2013). The Usage Control (UCON) method monitors the access continuously and supports attribute mutability; thus, when they change and do not satisfy the requirements anymore, the access is revoked, and the usage is canceled (Ouaddah et al., 2017). The UCON emerged when the technology evolution required trust management and Digital Right Management (DRM). Park and Sandhu (2004) introduced the $UCON_{abc}$, composed of eight components: subjects, subject attributes, objects, objects attributes, rights, authorizations, obligations, and conditions. The authorizations, obligations, and conditions are functional predicates that must be evaluated for usage decisions. The Context-Aware Access Control (CAAC) objective is to manage and control context-sensitive access to information and data resources in today's dynamic world (Kayes et al., 2020). In this case, the context means information about the state of an access control-specific entity (e.g., user, data resource, environment) or an associated relationship between entities.

Role-based policies: they control the access based on the activities the users execute in the system by using roles (Sandhu and Samarati, 1994). We can define *role* as a set of actions and responsibilities associated with a particular activity. Thus access authorizations on objects are specified for roles. The Role-Based Access Control (RBAC) is considered more scalable than the DAC and MAC models (Zhang et al., 2018). Kalam et al. (2003) introduced the Organization-Based Access Control (OrBAC), which considers the concept of an organization together with the concept of context to provide a framework for expressing the security policies of several organizations on centralized structures. The main components of OrBAC are organizations, subjects and roles, objects and views, actions and activities, security policy, contexts, and concrete authorization. The Trust-Based Access Control (TBAC) regulates access by evaluating trust values and classifying different trust degrees to protect resources (Almenárez et al., 2005). It focuses on open dynamic environments characterized by ad hoc networks and heterogeneous devices. The primary data elements of TBAC are users, agents, trust degree, resources, actions, and permissions.

2.2.3 Continuous Authentication

Authentication in computing is the process of identifying the various entities of a computer system with proof of identity. It is the method to prove their authenticity (Miraoui and El-Etriby, 2019). There are different types of authentication, which use different types of information and can be named factors. The authentication factors from Dias (2017); Miraoui and El-Etriby (2019); Cui et al. (2019) are: *Something You Know*, that involves memorized information (e.g., password, graphic pattern, response to a question, PIN code); *Something You Have*, which uses some possessed object (e.g., smart card, token, mobile device); *Something You Are*, based on physical characteristics of individuals and also known as biometrics (e.g., fingerprint or thumbprint, palm, handprint, retina, iris, voice, face); *Somewhere You Are*, which is related to user location (e.g., IP and MAC addresses, GPS); and *Something You Do*, based on user behavior or activity (e.g., gesture or touch, actions). The authentication can use only one method (single-factor) or more than one (multi-factor). It can also be multi-step when more than one factor is verified individually (Dias, 2017).

We can consider the following properties of authentication criteria as an identifier of a subject from Al-Naji and Zagrouba (2020): universality, every subject should possess at least one identifier; uniqueness, each subject should have a significantly unique and utterly different identifier; permanence, the identifier should perform in the same way over time irrespective of the matching criterion; storability, it must be possible to store the identifier; and simplicity, the identifier should be easy to collect. The IoT presents restrictions for the authentication process related to resources (e.g., memory, battery, processing). Thus the exchange and size of messages need to be reduced. Other challenges are the heterogeneity of devices and technologies, the scalability, and the privacy of identification data. Wazzeh et al. (2022) took advantage of the CA method to reinforce authentication to protect users' privacy and data security with behavioral biometrics, like voice and motion, and user profiling, like app preferences and behavioral profiles.

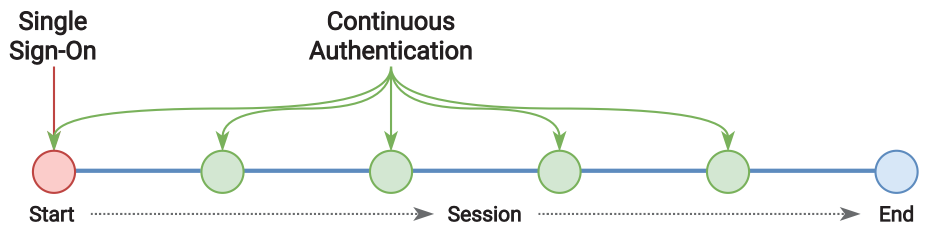


Figure 2.9: Single Sign-On and Continuous Authentication

Single Sign-On (SSO) is a user authentication technique that uses a single login credential to access multiple applications (Ashibani et al., 2019). The CA is the security mechanism to continuously monitor user actions by determining their authenticity (Al-Naji and Zagrouba, 2020). This approach minimizes impersonation attacks, detects sudden dangers, provides more convenience to users, and requires continuous collection and availability of information. As shown in Figure 2.9, the SSO authenticates the user only at the start of the session, presenting vulnerability if the attacker steals the session or the credentials, while CA re-authenticates the user several times to assure their legitimacy (Amraoui et al., 2020).

2.2.4 Intrusion Detection Systems (IDS)

The computing attacks pose significant threats to digital systems, causing disasters and compromising security properties, such as Confidentiality, Integrity, and Availability (CIA). The greater use of wireless technologies also increased systems vulnerabilities since they are much easier to attack than any wired network (Liao et al., 2013). Furthermore, the WSNs are characterized by multi-hop distributed operations, which make detecting, preventing, and locating attacks more complex (Alrajeh et al., 2013). An intrusion is any malicious activity inside the system by attackers to harm the network resources or sensor nodes (Alrajeh et al., 2013). According to Bace et al. (2001); Stavroulakis and Stamp (2010); Liao et al. (2013), the Intrusion Detection System (IDS) is the software or hardware system to automate the intrusion detection process.

It is difficult to prevent every attack in wireless networks as they are vulnerable; the IDSs are essential in detecting them to alert entities and mitigate the damages. The signature-based IDS is effective against well-known attacks, where their signature is maintained in a database and compared during the attack. Alternatively, the anomaly-based IDS detects new attacks by continuously monitoring traffic patterns or system activities (Alrajeh et al., 2013). The three IDS components are monitoring, which verifies locally and in the neighborhood, the traffic patterns, internal events, and resource utilization (Khan and Loo, 2009); analysis and detection, based on modeling algorithm, the network operations, behavior, and activities are analyzed and classified as normal or malicious; and alarm, which triggers warnings about the intrusion. The IDSs are passive, responsible for detecting attacks, and cannot take any preventive action (Alrajeh et al., 2013). The false positives (false alarms) and false negatives (undetected attacks) are the main measures of an IDS, which aims to minimize these values (Khan et al., 2010).

Based on Liao et al. (2013), we classify the IDSs proposals into five classes: statistics-based, pattern-based, rule-based, state-based, and heuristic-based. The statistics-based methods use predefined thresholds, mean and standard deviation, and probabilities to detect attacks. We can categorize them into statistics, distance-based, Bayesian-based, and game theory works. The pattern-based solutions use string matching to detect known attacks. We can divide them into pattern matching, perti net, keystroke monitoring, and file system checking works. The rule-based ones build the model using If-Then or If-Then-Else rules. We can group them into rule-based, data mining, model/profile-based, and Support Vector Machine (SVM) works. The state-based methods employ finite state machines derived from network behaviors. We can organize them into state-transition analysis, user intention identification, Markov process model, and protocol analysis works. The heuristic-based techniques apply bio-inspired computing and artificial intelligence. We can type them into neural networks, fuzzy logic, genetic algorithm, immune system, and swarm intelligence works.

According to Liao et al. (2013), we can classify IDS based on their technologies in five classes: Host-based IDS (HIDS), which monitors and collects the features for hosts containing sensitive information, servers running public services and suspicious activities; Network-based IDS (NIDS), which analyzes application activities and protocols to detect suspicious incidents by capturing network traffic; Wireless-based IDS (WIDS), which focuses in wireless network traffic, such as ad hoc networks, wireless sensor networks, and wireless mesh networks; Network Behavior Analysis (NBA), which detects attacks with unexpected network traffic flows; and Mixed IDS (MIDS), which employs multiple technologies being complete and more accurate.

We can organize the IDS time granularity to detect attacks as continuous, periodic, or batch runs. The detection time can be real-time (online) or non-real-time (offline). The detection response of an IDS can be passive when generating alarms only and active when taking corrective or preventive action (Liao et al., 2013).

2.2.5 Impersonation Attacks

In impersonation attacks, as shown in Figure 2.10, the adversary aims to assume the identity of a legitimate system entity by stealing their credentials or performing a man-in-the-middle attack (Adams, 2005). Impersonation attacks are also called masquerading (Komninos et al., 2014). According to Zheng et al. (2014), the system security must avoid an entity accessing the application using the credentials of another, define what authentic entities can access, and ensure the user's privacy. For that, the system should use secondary information to assure the entity's authenticity, like their behavior, context, and credentials more challenging to fake, such as fingerprint and face recognition. For instance, an attacker could steal the user's unique identifier that the smart hub generates during device registration to enable local access to access IoT resources (Geneiatakis et al., 2017). An eavesdropping attack is a common procedure for stealing authentic credentials attackers use. Alternatively, adversaries can also take advantage of a man-in-the-middle attack to send commands to the smart devices or even exploit previous requests in the replay attacks (Mocrii et al., 2018). Attackers usually perform impersonation attacks in the same network as they are. However, the cloud-based architecture for smart homes introduces the vulnerability for remote attacks when adversaries can reach the smart home router.

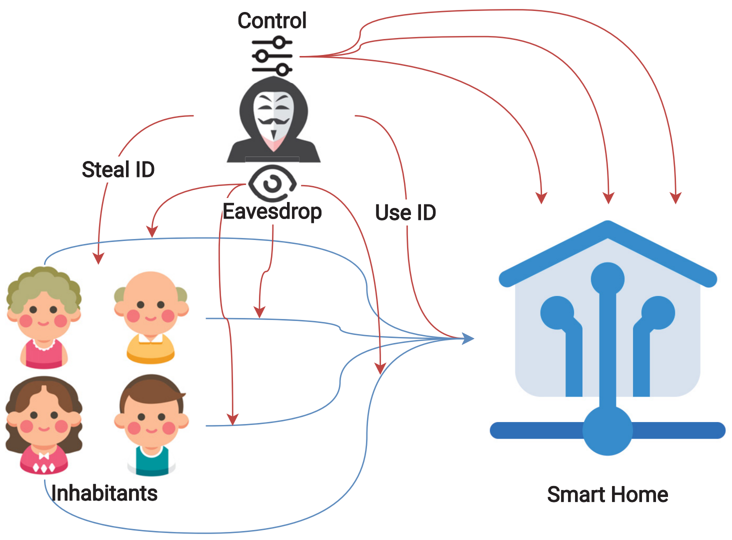


Figure 2.10: Impersonation attack in SHS

Continuous authentication minimizes the probability of a successful impersonation attack (Al-Naji and Zagrouba, 2020). As the SSO only verifies the user authenticity once at the beginning of the session, it opens a breach for impersonation attacks during the rest of the session. For instance, in De Fuentes et al. (2018), if the attacker can break the password of a smartphone, it could be blocked by analyzing the user behavior pattern between the legitimate user and the attacker. The CA improved security could also benefit applications such as autonomous vehicles or smart transportation by protecting against thieves, healthcare by keeping sensitive user data privacy, and military services by identifying suspects with drones (Al-Naji and Zagrouba, 2020). In smart homes, attackers could use impersonation attacks to control a smart lock and gain physical access to the house (Sikder et al., 2019a). Besides compromising confidentiality by accessing information from the system, integrity by performing actions and changes in the system, and availability by shutting down devices, impersonation attacks also affect non-repudiation, as attackers disguise their actions as performed by another user.

2.3 DECISION REASONING

Decision reasoning is the method of processing data to decide in real time depending on beliefs, plans, goals, and intentions (Georgeff and Ingrand, 1989). Traditionally, the decider module always reflected a mental model from its creator (Legrenzi et al., 1993), when given a set of the specific information system should act in one way or another. Recently, unsupervised learning techniques can reveal hidden patterns and information from the massive amount of data produced by IoT that support decision-making (Piccialli et al., 2020). Modern decision-making propositions also comprise trust-based models (Al-Hamadi and Chen, 2017) and fault-tolerant models (Gope et al., 2021). The following sections present common paradigms employed in the decision reasoning process, as Section 2.3.1 with context-aware, Section 2.3.3 with behavior-based, and Section 2.3.4 with Zero Trust (ZT).

2.3.1 Context-Aware

The context is the information utilized to characterize the state of an entity, which can be a person, place, physical, or computational object (Abowd et al., 1999). A system becomes context-aware when it uses the context to provide new information or services to the user. The main characteristics of a context-aware system are the presentation of information and services to a user, the automatic execution of a service, and tagging context to information for later retrieval. Abowd et al. (1999) presented the main challenges for context-aware computing as the development of a taxonomy and uniform representation of context types; infrastructure to promote the design, implementation, and evolution of context-aware applications; and discovery of compelling context-aware applications that assist our everyday interactions with ubiquitous computational services.

According to de Matos et al. (2017), we can categorize the context as primary, with direct information retrieved without any other operation (e.g., GPS sensor readings as location information). Ashibani et al. (2019) subdivided primary context or direct contextual information in user context, information about users, such as profile, calendar, social networks, and access patterns; device context, information related to devices, such as location, current, and voltage values, Wi-Fi access points, operating systems, and running/installed applications; network data, like IP address, MAC address, link speed, ping times, and trace routes; and environmental context, information related to the physical environment, such as temperature, weather, lighting, loudness, or humidity. The secondary context refers to indirect contextual information that the system computes using primary context (de Matos et al., 2017), for example, calculating power consumption using voltage and current values or speed from multiple GPS locations. Other classifications are static and dynamic and internal and external. Ashibani et al. (2019) considered other classifications as static contextual information, which changes very slowly or does not change at all (e.g., address, person name), and dynamic contextual information, which changes over time (e.g., time, person location); internal contextual information, retrieved from the devices used by the user (e.g., battery level, current and voltage readings), and external contextual information, obtained from external resources (e.g., location obtained from GSM operator).

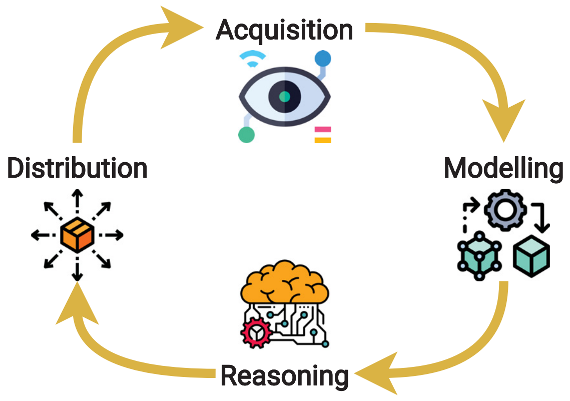


Figure 2.11: Context information life cycle

The context life cycle includes acquisition, modeling, reasoning, and distribution (de Matos et al., 2017) as presented in Figure 2.11. We can organize the context acquisition phase by responsibility (Pietschmann et al., 2008), as push when sensors push data to the consumer or pull when the consumer request data from the sensors. The frequency can be instant when events occur instantly or interval when events span a specific period. The possible sources are physical sensors (hardware), middleware infrastructure, or context servers (e.g., databases, web services) (Chen et al., 2004). As specified by Indulska et al. (2003), we can group the sensors into: physical, which generates data by themselves (e.g., temperature, humidity, microphone, touch); virtual, which collects data from many sources and publish it as sensor data (e.g., calendar, contact number directory, Twitter statuses, email, chat); and logical (software), that combines physical and virtual sensors to provide more meaningful information (e.g., weather information). The acquisition processes are through sensation, as the data sensed by the sensor (e.g., temperature from the sensor); derivation, which results from computations on sensor data (e.g., the distance between two points using GPS coordinates); or manually input, provided by users, such as preferences (e.g., the user does not like to receive notifications between 10 pm and 6 am).

Context modeling comprehends information defined by attributes, characteristics, and relationships, which is validated and grouped in a context information repository (Bettini et al., 2010). Each modeling technique of a context presents its strengths and weaknesses. According to Chen and Kotz (2000); Strang and Linnhoff-Popien (2004), they are classified in: key-value, where each data has a key, being the simplest form of representation, though not scalable and suitable to store complex data structures; markup scheme, which stores data within tags, like XML, allowing efficient data retrieval, albeit not providing advanced expressive capabilities; graphical, that models context with relationships, such as Unified Modelling Language (UML), Object Role Modelling (ORM), SQL database and NoSQL database, which are easy to use and optimized techniques, but the different implementations compromises the interoperability; object-based, that uses class hierarchies and relationships, promoting encapsulation and reusability, though its validation is complex due to the lack of standards and specifications; logic-based, represented by facts, expressions and rules, providing a rich expressiveness, but possibly complicated to be adopted; and ontology-based, where context is organized in ontologies with semantic technologies, which yields a wide range of development tools and reasoning engines, whilst it is computationally intensive and time consuming for high amount of data.

Context reasoning or inference is the process of providing high-level context deductions from a set of contexts (Guan et al., 2007). According to Nurmi and Floréen (2004), its phases are: preprocessing, which cleans data by filling missing values, removing outliers, validating context, etc.; data fusion, which combines multiple sensors data to produce more accurate, complete, and dependable information (Hall and Llinas, 1997); and inference, that generates high-level information using lower-level context. Following Perera et al. (2014), we categorize the reasoning into: supervised learning, which learns from labeled data; unsupervised learning, which can find hidden structures in unlabeled data; rules, being the classic If-Then-Else, the most straightforward way to model human thinking and reasoning in machines; fuzzy logic, which allows approximate reasoning with partial truth as acceptable; ontology-based, which uses description logic to represent knowledge with formalisms; and probabilistic logic, that considers probabilities attached to the facts related to the problem to understand the occurrence of events. The distribution of a context is responsible for delivering context to the consumers (de Matos et al., 2017) and, as specified by Perera et al. (2014), can be: query, where the consumer requests the information; or subscription (publish/subscribe), where the consumer subscribes to receive specific information periodically or when an event occurs.

2.3.2 Ontology Modeling

The current globalized world enables cooperation and collaboration of multidisciplinary people towards the progress of many areas. Communication between people, organizations, and software systems can only be effective when every entity understands a common language. The agreement on the terms and meanings shared by a group is called ontology (Roche, 2003). The epistemological definition of ontology embraces three dimensions: *knowledge or conceptualization* for understanding the world, *language* to speak about the world, and *logic or representation* for the manipulation of our understanding. We can use ontologies for communication between people and organizations; interoperability between systems; system engineering, specification, reliability, and reusability; knowledge management; and natural language treatment for semantic analysis and lexical structure. An ontology can be classified in terms of formality as highly informal when definitions are expressed in natural language; semi-informal, when practiced to reduce ambiguity with a restricted and structured form of natural language; semi-formal, if the definitions are described in an artificial formally defined language; and rigorously formal, when the definitions are precisely defined with a formal semantics. Furthermore, we group them by the type of knowledge in: generic or top ontology, with general concepts (e.g., time, space, etc.); domain-based, dedicated to a particular domain (e.g., chemical, medicine, etc.); application-based, specific to a particular task for an application; and meta-ontology or representation-based, with the knowledge representation principles to define concepts of domain and generic ontologies (Roche, 2003).

According to Hitzler (2021), it usually envisions the Semantic Web as an enhancement of the current World Wide Web with machine-understandable information (as opposed to most of the current Web, mainly targeted at human consumption), together with services (intelligent agents) utilizing this information. In the Semantic Web context, ontologies are the primary vehicle for data integration, sharing, and discovery, and the driving idea is that ontologies should be reusable by others. There were efforts in the early 2000s to come up with ontological patterns. The DARPA Agent Markup Language (DAML) program ran from 2000 to 2006, and the OIL language from the European Union-funded On-To-Knowledge project happened between 2000 and 2002. They merged to form the Web Ontology Language (OWL) W3C standard in 2004, which later became OWL 2 (Hitzler et al., 2009b) in 2012. OWL, at its core, is based on description logic, that is, on a sublanguage of first-order predicate logic using only unary and binary predicates and restricted use of quantifiers, designed in such a way that logical deductive reasoning over the language is decidable (Hitzler et al., 2009a). The RDF is another W3C standard syntax for expressing directed, labeled, and typed graphs (Guus Schreiber, 2014).

The linked data emerged as a significant driver in the Semantic Web area from 2006 until the early 2010s (Hitzler, 2021). It consists of a (by now rather large) set of RDF graphs linked because many IRI identifiers in the graphs also appear in other, sometimes multiple, graphs. During the Linked Data era, ontologies played a much less prominent role. They often were schemas to inform the internal structure of RDF datasets. However, the information in RDF graphs in the Linked Data Cloud was shallow and relatively simplistic compared to the overpromises and depth of research from the Ontologies era. The knowledge graphs term appeared in 2012 from Google as usually understood to be much more internally consistent and tightly controlled artifacts (Hitzler, 2021). Consequently, it challenged the value of external links (that is, to external graphs without tight quality control). In contrast, it highlights the quality of content and the underlying schema.

According to YE et al. (2007), the criteria needed to assess ontologies are clarity, as terms must be identified unambiguously and communicated effectively; coherence, as definitions must be consistent without conflicts; ontological commitment, as they should make just enough claims about the domain to support the intended knowledge sharing and reuse; encoding bias, as they should be specified at the knowledge level without depending on a particular symbol-level encoding; and extensibility, as they should offer a conceptual foundation for anticipated and potentially anticipated tasks. Furthermore, some guidelines should be followed to build ontologies: requirement analysis, as it is needed to specify the purpose of the ontologies and their necessity; building ontologies, as the developers should arrange the concepts and terms that need to be captured by these ontologies; evaluation, as it is required to perform assessments of the ontology throughout the whole life cycle of ontology development; and documentation to describe the completed ontologies.

2.3.3 Behavior-Based

Activity and anomaly detection in smart homes is a complex process. The inhabitant's activities can be continuous or interleaved when a person pauses an activity to start another and then returns to the previous one (Bakar et al., 2016; Hu and Yang, 2008). They can also be concurrent when a person does more than one task simultaneously. The temporal-ordered nature of sensory data makes the activity classification problem the same as the Sequence Classification (SC) problem. As specified by Bakar et al. (2016); Xing et al. (2010), the three general types of SC are: feature-based, such as k-gram and pattern-based feature selection; sequence distance-based, such as KNN and SVM; and model-based, such as Naive Bayes, Markov Model, Hidden Markov Model (HMM). The sensory data can be analyzed at the lower sensory level when data collected from sensors are classified as activities, going through preprocessing, segmentation, and feature extraction. According to Bakar et al. (2016), the activity classification (modeling) can be divided into supervised Activity Recognition (AR), which is the process of mapping predefined activity classes to the sensory reading generated data (Cook et al., 2012) and can involve algorithms as Decision Tree (DT), Neural Network (NN), Instance-Based learner (IB) and SVM; and unsupervised Activity Discovery (AD), when the knowledge or desired data pattern is a discovery by finding the frequency of data co-occurrence in generated sensory reading (Cook, 2007; Cook et al., 2012) and involve techniques as If-Then rules, Episode Discovery (ED) (Heierman and Cook, 2003), and Greedy Search (Cook, 2007).

On the other hand, systems analyze the sensory data at the higher activity level when the output is the inhabitant's activities, labeled as sleeping, eating, showering, entering, leaving home, etc. The activities can be further analyzed at this level to infer high-level information, like the inhabitant's lifestyle. Anomaly detection detects behavioral anomalies of inhabitant activities on unusual data (Bakar et al., 2016). The system can inspect the behavior in different contextual aspects, such as spatial (location), temporal (time and duration), order of time, order of activity, health status, etc. The two methods of detecting behavioral changes are profiling, which creates a standard behavior model and looks for deviations; discriminating, which learns from previous anomaly data. The profiling is more realistic as anomaly data is rare in real life, compromising the learning process of the discriminating application (Cardinaux et al., 2008). The three ways for anomaly decision from Bakar et al. (2016) are: point anomaly, which decides by a predefined threshold value; contextual anomaly, based on contextual factors (e.g., temporal, spatial); and collective anomaly, which defines data instance as an anomaly in a collection that could not be noticed anomalous individually.

The algorithms for anomaly detection from Bakar et al. (2016) are: the histogram, that captures the frequency of occurrence an its location, though is not suitable for independent data, cannot capture data dependency structure and relate two attributes; the Gaussian Mixture Model (GMM), which can relate two data attributes for activity classification, but the detection efficiency depends on the matching times; the Hidden Markov Model (HMM), that is simple, efficient for sequential data, have temporal dependency structure, and can handle noisy data, however have the "cold start" issue, need full description of the big data, require lots of trainings, has the conditionally independent assumption, and requires human intervention that may cause label biased problem; the conditional random field, that have no "cold start" problem, can capture long range dependency data structure, and can handle noisy data, though have expensive training cost for long-range dependency data structure, requires human intervention that may cause label biased problem, and requires anomaly data instance; the artificial neural network, which is adaptive to new information rules and efficient for typical problem of neural networks, but cannot incorporate new user added rules, has complex architecture, and not human readable; SVM, which is efficient for linearly separable data, however requires anomaly data instance; the semantic rule, that can reduce false positive rate and provide human readable logic rules, though does not handle noisy data; and binary similarity measure functions, which are easy to compute, but only takes binary data, is not suitable for high-level activity data, and cannot capture type, when and where anomalies happened.

The Markov Chain is a discrete-time stochastic process that denotes a set of random variables and defines how these variables change over time (Sikder et al., 2020). The two main assumptions of the Markov Chain model are: the current state t only depends on the previous state $t-1$; the transition between state $t-1$ and t is independent of time. Section 2.3.3 represents the probabilistic condition the Markov Chain imposes. Sikder et al. (2020) proved that Markov Chain is the best model to detect anomaly behaviors in a context-aware system. For instance, considering binary sensors, the model has $m = 2^n$ states for a number of n sensors. The probability from state i to state j is denoted by p_{ij}, and Equation (2.4) expresses the Markov Chain transition matrix.

$$\begin{aligned} P(X_{t+1} = x | X_1 = x_1, X_2 = x_2, \dots, X_t = x_t) = P(X_{t+1} = x | X_t = x_t), \\ when,\ P(X_1 = x_1, X_2 = x_2, \dots, X_t = x_t) > 0 \end{aligned} \tag{2.3}$$

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots & \cdots & p_{1m} \\ p_{21} & p_{22} & p_{23} & \cdots & \cdots & p_{2m} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ p_{m1} & p_{m2} & p_{m3} & \cdots & \cdots & p_{mm} \end{bmatrix} \tag{2.4}$$

2.3.4 Zero Trust (ZT)

The traditional security approach depends on a well-monitored perimeter that keeps attackers outside the network. However, it is hard to define boundaries around networks on new pervasive technologies in the IoT paradigm; thus, attackers gain access to all resources when they get in. Kindervag (2010) noticed that and coined the Zero Trust (ZT) term referring to a novel paradigm focused on verifying the entity's authenticity and trust continuously instead of the usual static approach to protecting resources. According to Scott W. Rose, Oliver Borchert, Stuart Mitchell (2020), ZT provides a collection of concepts and ideas designed to minimize uncertainty in enforcing accurate, least privilege per-request access decisions in information systems and services in the face of a network viewed as compromised. Meanwhile, Zero Trust Architecture (ZTA) is an enterprise's cybersecurity plan that utilizes ZT concepts and encompasses component relationships, workflow planning, and access policies.

The main issue ZT aims to minimize is the insider attackers, which are usually in a position of trust. Further, it should not have any trusted entity or interface by default. A security system must not trust any entity but verify and assure its intentions. In a network, the data flow verification depends on the information derived from packets. From the previous assertions, Kindervag (2010) proposed the following concepts of ZT: ensure that all resources are accessed securely regardless of location; adopt a least privilege strategy and strictly enforce access control; and inspect and log all traffic. Briefly, ZT flips the mantra "trust but verify" into "verify and never trust". The system must ensure that the subject is authentic and the request valid, so ZT involves authentication and authorization. According to Scott W. Rose, Oliver Borchert, Stuart Mitchell (2020), the ZT tenets are: all data and computing services are considered resources; all communication is secured regardless of network location; access to individual enterprise resources is granted on a per-session basis; access to resources is determined by dynamic policy, including the observable state of client identity, application/service, and the requesting asset, and may involve other behavioral and environmental attributes; the enterprise monitors and measures the integrity and security posture of all owned and associated assets; all resource authentication and authorization are dynamic and strictly enforced before access is allowed; and the enterprise collects as much information as possible about the current state of assets, network infrastructure, and communications and uses it to improve its security posture.

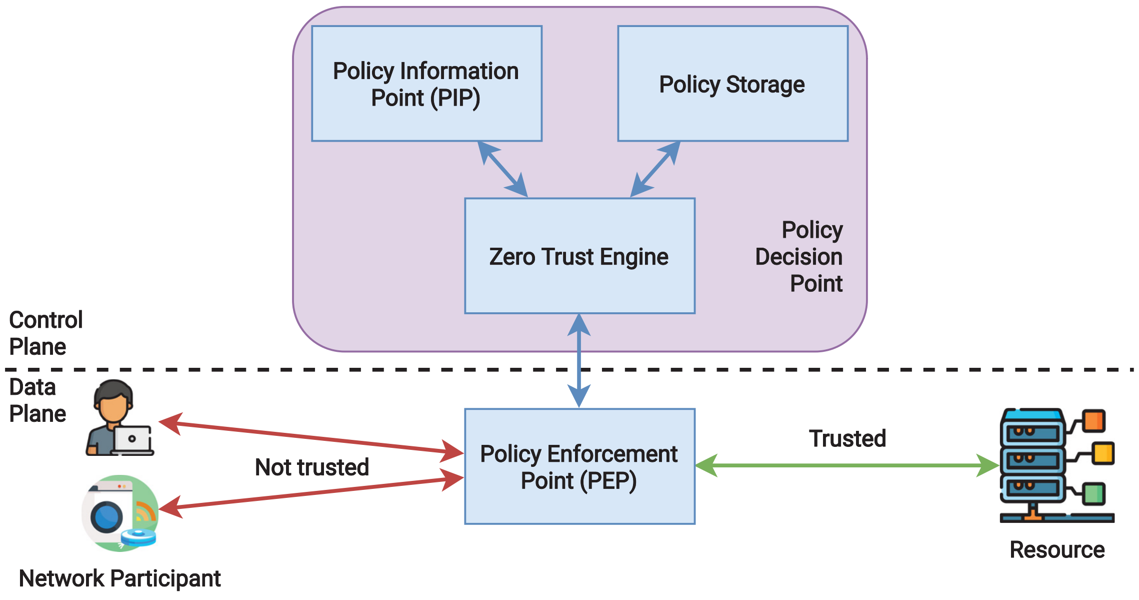


Figure 2.12: Zero Trust Architecture

Figure 2.12 represents ZTA based on Scott W. Rose, Oliver Borchert, Stuart Mitchell (2020); Buck et al. (2021). The Policy Decision Point (PDP) is composed of the ZT Engine or Policy Engine (PE), which is responsible for granting or revoking subject access to resources based on the information provided by the Policy Information Point (PIP) and rules stored on the Policy Storage. Every request reaching the Policy Enforcement Point (PEP) is not trusted and verified to give access. PEP enables, monitors, and eventually terminates connections between a subject and a resource. The PE uses a Trust Algorithm (TA) to calculate the variables and decide the access. The inputs of TA from Scott W. Rose, Oliver Borchert, Stuart Mitchell (2020) are: access request, subject database, asset database, resource requirements, and threat intelligence. TAs can be criteria-based when it assumes a set of qualified attributes that must be met before the system grants access to a resource or an action (e.g., read/write) is allowed, or score-based when it computes a confidence level based on values for every data source and configured weights. We also organize TA as singular, which treats requests individually, or contextual, which uses recent history to evaluate access requests. Although singular is faster and simpler, contextual can detect more complex attacks. Finally, the network requirements to implement ZTA are: assets have basic network connectivity; distinguishable assets between owned or managed and the current security posture of devices; observe all network traffic; resources should not be reachable without accessing a PEP; the data plane and control plane are logically separate; assets can reach the PEP component; the PEP is the only component that accesses the PDP as part of a business flow; Remote assets should be able to access resources without needing to traverse network infrastructure first; the infrastructure used to support the ZTA access decision process should be made scalable to account for changes in process load; assets may not be able to reach certain PEPs due to policy or observable factors.

2.4 DECISION COMPUTING

The authorization system can compute the decision locally using edge computing or a remote server via cloud computing. The IoT applications might require very short response times, sometimes involving private and vast amounts of data (Shi and Dustdar, 2016). Although the cloud provides more computing power than devices at the network edge, it still depends on the network's bandwidth to transfer data. As edge devices produce more data over time, the network becomes the cloud computing's bottleneck (Shi and Dustdar, 2016). Edge computing yields shorter response times, more efficient processing, and less pressure on the network. Besides, users lose control over their data by storing it in the cloud, facing privacy leakage risks (Khan et al., 2019). The following sections present the definitions of cloud computing and edge computing.

2.4.1 Cloud Computing

Mell and Grance (2011) defined cloud computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model comprises five essential characteristics, three service models, and four deployment models. The essential characteristics are on-demand self-service, the consumer can automatically adjust computing parameters such as server time and network storage without requiring human integration; broad network access, interoperability by using standard mechanisms to promote use by heterogeneous client platforms; resource pooling, the provider provisions the resources (e.g., storage, processing, memory, network) to several consumers on demand; rapid elasticity, capabilities need to be provisioned and released to scale rapidly; measured service, resources usage are monitored, controlled and reported to provide control, optimization, and transparency. The service models are: Software as a Service (SaaS), consumers can use the provider's applications running on a cloud infrastructure; Platform as a Service (PaaS), consumers can deploy onto the cloud infrastructure using limited by provider languages, libraries, services, and tools; Infrastructure as a Service (IaaS), consumers can use the cloud infrastructure to deploy and run arbitrary software, including operating systems and applications. The deployment models are private cloud, provisioned for exclusive use by a single organization; community cloud, provided for exclusive use by a community of consumers with shared concerns; public cloud, provisioned for open use by the general public; hybrid cloud, a composition of two or more of the other models.

2.4.2 Edge Computing

Edge computing brings the services to be executed in the network edge by the devices at some specific location. The main characteristics of edge computing are the high bandwidth, ultra-low latency, and real-time access to the information (Khan et al., 2019). It minimizes the load to the cloud and reduces latency by bringing the services close to the end users. The dense geographical distribution of edge computing facilitates location-based mobility services, big data analytics speed and accuracy, and real-time analyses on a large scale. The Locator ID Separation Protocol (LISP) enables mobility support, which decouples the location identity from the host identity. The location-awareness permits mobile users to access services to the edge server closest to their physical location using a cellular network, GPS, or wireless access points. The proximity between users and the edge servers improves availability, user experience, resource allocation, and services decision by extracting network context information and analyzing user behavior. The low latency attribute allows users to execute resource-intensive and delay-sensitive applications on the edge devices (e.g., router, access point, base station, dedicated server). Context information edge computing can enhance user satisfaction and quality of experience. The heterogeneity of technologies, software, hardware, and protocols poses a challenge to the interoperability of services in edge computing (Khan et al., 2019).

The main edge computing models are cloudlets, fog computing, and mobile edge computing. Cisco (2015) introduced the concept of fog computing to connect and analyze data from thousands and millions of different kinds of things spread over large areas. A cloudlet is a trusted, resource-rich computer or cluster of computers well-connected to the Internet and available by nearby mobile devices (Satyanarayanan et al., 2009). Mobile Edge Computing (MEC) was introduced to bring the cloud services and resources closer to the user proximity by leveraging the available resources in the edge networks (Ahmed and Rehmani, 2017). These models aim to bring processing and storage closer to the users. The objectives of edge computing are to minimize latency, optimize cost, and manage network, energy, resources, and data (Hassan et al., 2018).

IoT devices present restrictions in processing, storage, and energy. Thus, edge computing in IoT depends on edge servers to augment their capabilities (Hassan et al., 2018). Computing nodes comprehends servers, base stations, routers, and vehicles, providing more IoT resources. Smart homes have several devices and can benefit from edge computing to monitor and measure the house, like utilities (e.g., water, electricity, gas), to provide real-time data analytics. According to Hassan et al. (2018), the role of edge computing in the IoT impacts the data acquisition, as data is obtained and processed at the same location; inferential controls, which is possible as edge devices have more contextual information than the cloud; data analysis in real-time with low latency, reduced cost, and network load since it does not need to go back and forth from the cloud; decision making locally for time-sensitive applications; and enhanced data security, once it does not involve extensive routing, data is less prone to be intercepted.

2.5 SUMMARY

This chapter presented the fundamental theories and concepts to understand and support this work. The IoT is already an established paradigm evolving to be even more pervasive and ubiquitous, as in CPSS and IoE. Besides, the smart home application has great potential to make the life of millions more comfortable. However, there are many challenges related to technical and social aspects, including resistance linked to privacy concerns. The security in IoT is gaining proportional importance to the popularity of IoT since the pervasive characteristic also contributes to increasing the attack surface. Privacy is one of people's most significant concerns in this digital era, and impersonation attacks pose severe risks. We can employ many approaches to avoid attacks, such as access control, continuous authentication, and IDS. Furthermore, they are supported by decision reasoning, such as context-aware, behavior-based, and ZT methods, and decision computing, such as cloud computing and edge computing models.

3 STATE-OF-THE-ART OF SECURITY IN THE SMART HOMES

This chapter broaches the security state-of-the-art in smart homes, including the main threats and their countermeasures, followed by the strategies against impersonation attacks. Section 3.1 exposes an overview of the approaches to securing a smart home. Section 3.2 describes the current works on smart homes access control. Section 3.4 discourses about the current continuous authentication works in smart homes. Section 3.3 introduces the recent works on smart homes Intrusion Detection Systems (IDSs). Finally, Section 3.5 presents the discussion on the exposed works by identifying the requirements of the smart homes and selecting the comparative works to analyze limitations and directions to follow.

3.1 MAIN THREATS AND COUNTERMEASURES

The evolution of technology and the popularization of smart devices contribute to broader access to smart homes, which presents challenges such as device heterogeneity, communication heterogeneity, technical expertise, resource constraints, data collection, and security and privacy (Panwar et al., 2019). Although cheaper smart devices are emerging, their security is weak (Kaspersky, 2021), leading to vulnerabilities in smart homes. It raises inhabitants' concerns with their data and physical privacy as the smart homes are Cyber-Physical Social Systems (CPSS). The main threats against the security of the smart homes, detailed in Table 3.1, are: eavesdropping, also known as passive information gathering, which is the interception of information about inhabitants and their behavior compromising the confidentiality, that is more prone to happen in wireless networks with a possible countermeasure being the link layer encryption (Butun et al., 2020); impersonation, when attacker use stolen credentials of an authentic user to exploit the system compromising the confidentiality, integrity, and non-repudiation with continuous authentication as one of the countermeasures (Al-Naji and Zagrouba, 2020); software exploitation, which is caused by negligence of users not taking basic security measures compromising the confidentiality and integrity with the remedy being the correct update and configuration of the software (Mocrii et al., 2018); Denial of Service (DoS), which can be explored in several network layers, characterized generically by the overload of the network resources to make them inaccessible, compromising the availability with one possible countermeasure as limiting the total number of connections (Butun et al., 2020); and ransomware, which is relatively recent type of attack, where valuable information is encrypted and a ransom is demanded in exchange of the secret key, compromising the integrity and availability, being access control a prevention for it (Al-rimy et al., 2018).

Security in smart homes comprises techniques associated with preventing, detecting, and mitigating Byzantine attacks. The following sections review techniques to prevent malicious attackers from breaking into the system, detect when the attacker has succeeded in the invasion, and mitigate presumable damage from this action. Access to a smart home must be restricted to its inhabitants, as detailed in Section 2.2.2, being authentication a vital phase to identify who is requesting actions and information from the system. The smart home environment is dynamic regarding both devices and users. Continuous Authentication (CA) related works aim to identify users interacting with the system throughout the session rather than relying on a unique authentication at the start of the session, as described in Section 2.2.3. Furthermore, the Intrusion Detection Systems (IDS) proposals efficiently detect invasions, as expressed in Section 2.2.4, and have been applied in smart homes. Generally, the CPSS, besides including context information collected from the environment, also reckons human behavior to make a decision, as people participate directly in the system. That can be verified in the following sections as they present at least one work based on behavior and context.

Table 3.1: Main threats against smart homes security

Attack	Layer	Description	Properties Compromised	Countermeasures
Eavesdropping	All	Intercept information about inhabitants and their behavior	Confidentiality	Link layer encryption, SensorWare communication multicast model, Key pre-distribution (Butun et al., 2020), Encryption and authentication (Kuyucu et al., 2019)
Impersonation	Application	Use stolen credentials of an authentic user to exploit the system	Confidentiality, Integrity, Non-repudiation	Continuous authentication (Al-Naji and Zagrouba, 2020), Intrusion Detection Systems (Sikder et al., 2019a), Authorization (Ghosh et al., 2019)
Software exploitation	Application	Exploitation of breaches caused by the negligence of users not taking basic security measures	Confidentiality, Integrity	Update and configure software (Mocrii et al., 2018), Advise users (Kuyucu et al., 2019)
Ransomware	Application	Encrypt information and demand for ransom to provide the secret key	Integrity, Availability	Prevention, detection, and prediction (Al-rimy et al., 2018)
Denial of Service (DoS)	All	Overload the network resources to make them inaccessible	Availability	Limiting the total number of connections (Butun et al., 2020)
Jamming DoS	Physical	-	-	Spread-spectrum communication, JAM (re-routing), Wormhole technique, Swarm intelligence, JAM (mapping) (Butun et al., 2020)
Link Layer Flooding	Link	-	-	Anomaly detection on motes (Butun et al., 2020)
HELLO-flooding	Network	-	-	Bidirectional verification technique, Identity verification protocol, Multi-path multi-base station routing, μ-TESLA (Butun et al., 2020)
SYN-flooding	Transport	-	-	SYN-cookies, Client puzzles (Butun et al., 2020)
Path-based DoS	Application	-	-	One-way hash chains (Butun et al., 2020)

3.2 AUTHORIZATION

This section discusses the security works in smart homes related to authorization and their different propositions. In Rahmati et al. (2018), it is proposed a risk-based permission model to control the access to devices in a smart home that regards the risk asymmetry between functionally-related operations in a smart home creates an imbalance between the level of access to a device that an app needs. Though, this approach can violate the principle of least privilege, an important concept in computer security. Moreover, to define the permissions policy, it classifies operations risk as low, medium, and high using a dataset with user-perceived risks to group similar operations. It reduces access to high-risk operations by 60% by enforcing this risk-based model in SmartThings. The study outlined the benefits of using this risk-based classification to define the permissions policy. Nevertheless, it depends on smartphone usage and does not reckon any other property for access control, leading to possible restrictions on authentic users.

Ghosh et al. (2019) presents SoftAuthZ Framework, as shown in Figure 3.1, a context-aware behavior-based authorization framework for smart homes. Among the work contributions is modeling expected belief on device access requests based on the user's past request patterns and the request context. It organizes the devices in classes reflecting their general accessibility to the users. The less restrictive is the General_purpose, which includes mostly regular devices, such as lights, plugs, ovens, etc. The following is the Controller class, which consists of devices capable of sending control signals, like voice assistants and hub devices. The most restrictive class is Safety and Security, which contains devices for surveillance and security purposes, such as locks and cameras. A list of capabilities is assigned to each class, representing actions that users can perform with devices. It also organizes the requester in classes, from the least restrictive to the most: Owner, Spouse, Family, Children, and Guest. The context information analyzed regards the location of the requester (Home, Premises, Outside), time of request (Conventional, Odd), users around the requester (Present, Absent), and requester's age (adult, teenage, child). The work mainly tries to block insider attacks by setting a confidence threshold to execute a request. However, the dependence on historical data, as the other works focused on Anomaly Detection, poses an issue of cold start, as the system needs previous training (Bakar et al., 2016).

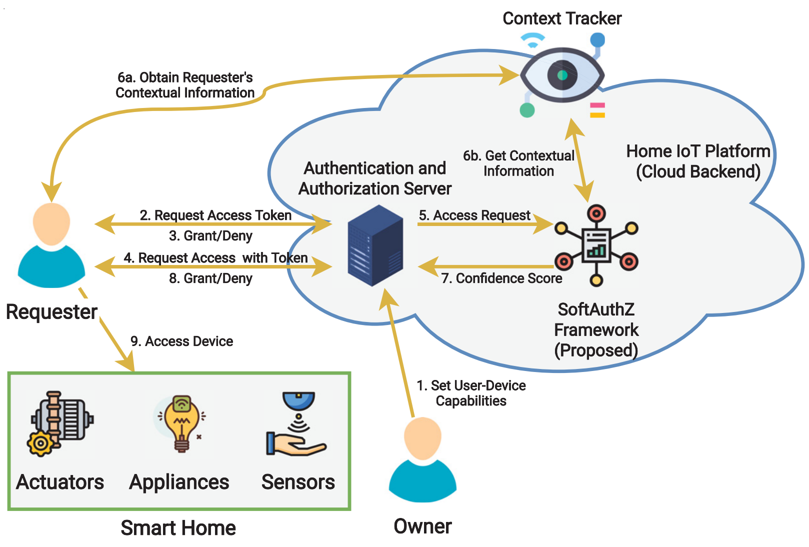

Context Tracker
6a. Obtain Requester's Contextual Information
Authentication and Authorization Server
6b. Get Contextual Information
Home IoT Platform (Cloud Backend)
2. Request Access Token
3. Grant/Deny
4. Request Access with Token
8. Grant/Deny
5. Access Request
7. Confidence Score
Requester
SoftAuthZ Framework (Proposed)
9. Access Device
1. Set User-Device Capabilities
Actuators
Appliances
Sensors
Smart Home
Owner

Figure 3.1: SoftAuthZ Framework architecture (Ghosh et al., 2019)

Table 3.2: Authorization works in smart homes

Work	Approach	Contributions	Limitations
Tyche: A Risk-Based Permission Model for Smart Homes (Rahmati et al., 2018)	Auth based in risks	Grouping of similar operations; risk-based permissions; survey on perceived risk by users	Absence of constant user verification; lack of fine-grained policy configuration
SoftAuthZ: A Context-Aware, Behavior-Based Authorization Framework for Home IoT (Ghosh et al., 2019)	Auth based on behavior and context	Soft-security metrics like trust and belief; use of various behavioral and contextual attributes	Dependence on historical data; cold start issue for new users
Kratos: Multi-User Multi-Device-Aware Access Control System for the Smart Home (Sikder et al., 2019b)	Auth for multi-user and multi-device conflict resolution	Supports multiple users and devices; automatic conflict negotiation; flexible, user-controlled policies	Absence of constant user verification
Enterprise Security with Adaptive Ensemble Learning on Cooperation and Interaction Patterns (Quintal et al., 2020)	Auth based on behavior and context	Contextual metrics arising from RBAC system interactions (shareability, evaluation, cooperation); Auth for business (complex) environments	Dependence on historical data; cold start issue for new users
Context Sensitive Access Control in Smart Home Environments (Dutta et al., 2020)	Auth based on behavior and context	Dynamic Auth based on user context, information collected by cloud providers and device type; anomaly detection to provide feedback for users	Complex configuration for inexperienced users

KRATOS, described by Sikder et al. (2019b), presents a novel multi-user and multi-device-aware access control mechanism that allows smart home users to specify their access control demands flexibly. The policy manager automatically resolves conflicts of user's demands, determined using the interaction module and translated into access control policies in the backend server. Its flexibility, automatic resolution, and mobile app interface make the system more user-friendly, maximizing users' will to keep using it. KRATOS detected with a success rate of 100% and without significant overhead five different threats, such as overprivileged controls, privilege abuse, privilege escalation, unauthorized access, and transitive privilege. However, it assumes that users are authentic; thus, an impersonation attack could be successful. In Quintal et al. (2020), enterprise-focused access control is introduced based on social and contextual properties. The decision process regards features arising from the interaction of users with documents. The shareability feature measures how much users share a document. The valuation feature quantifies the value of a document itself. The user cooperation feature represents actions performed on common documents by two users. The work uses an ensemble learning method with an adaptive weighted-voting technique to calculate the trust score to control access. Although the proposed system has increased security compared to other classifiers, it requires historical data for training and could impact the latency in practical real-time usage.

In PALS, detailed by Dutta et al. (2020), context-based access control for smart homes is proposed, where the decision process depends on the user context, details of the information collected by the cloud service provider, and device type. PALS uses data from the physical sensors to infer the context and ABAC rules configured with Semantic Web Rule Language (SWRL) (Horrocks et al., 2004). The context consists of the activity, which can be household, leisure, or work; the identity, which can be a child or adult family member, or stranger; the location, which can be a building, city, country, county, or place (home, private room, public room, or semi-private room); and the time, which can be instant or interval (downtime, off hours, weekday, weekend, or working hours). Knowledge Graph (KG) supports PALS fed by the context and the device's states. The KG defines the access control policies; then, PALS performs semantic reasoning to decide whether action should be granted. It also has a behavioral anomaly detection supported by Hidden Markov Model (HMM) to alert users via smartphone notifications. It provides flexible tools to build complex ontologies, though they are complex to configure for inexperienced users.

Dimitrakos et al. (2020) proposes the UCON+ as an extended Usage Control (UCON) method to include trust evaluation to perform continuous authorization. UCON is a framework created to focus on Digital Rights Management (DRM) and is suitable to monitor the execution rights of a subject over a resource continuously. UCON+ employs a Zero Trust Architecture (ZTA), which requires continuous permissions and authorization policies verification. Zero Trust (ZT) is an evolving set of cybersecurity paradigms that move defenses from static, network-based perimeters to focus on users, assets, and resources (Scott W. Rose, Oliver Borchert, Stuart Mitchell, 2020). It assumes any entity can become malicious, regardless of its past authenticity and reliability, so trust should be evaluated continuously. Although UCON+ is highly customizable, it is unsuitable for inexperienced users, as an SHS Auth should be.

3.3 INTRUSION DETECTION SYSTEMS

This section discusses the existing security works in smart homes based on the IDS approach. Aegis, exposed in Sikder et al. (2019a) and shown in Figure 3.3, is a context-aware IDS that relies on a correlation between user activities and devices. Figure 3.2 illustrates that as the user performs activities, the sensors and devices react in a specific pattern, which can check for malicious actions. The system extracts features from sensors, devices, controller devices, and smart apps. Then, they generate the context array, representing the smart home state at that instant. The anomaly detector depends on a Markov Chain that expresses the probabilities of transitioning from one state to another. For example, in Figure 3.2, transitioning from sub-context 1 to sub-context 2 is valid as the user can perform this activity. However, transitioning from sub-context 1 to sub-context 4 is impossible, considering it must follow the sequence through sub-context 2 and 3. Besides the detection mode, which notifies malicious activities, Aegis also counts on an adaptive training mode, which collects user feedback by confirming the suspect and retraining the model. Although the work proves to cope with different home layouts and inhabitant numbers adaptively, it needs a large amount of data to feed the anomaly detector module for training. Additionally, it depends on cloud computing, which brings a vulnerability for a smart home since an attacker could eavesdrop on the network to steal data or cut the Internet means to isolate the house.

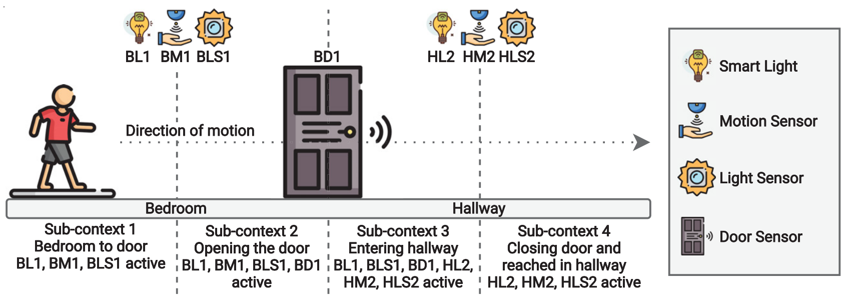


Figure 3.2: User activity-device correlation (Sikder et al., 2019a)

Pan et al. (2019) details other context-aware IDS enabled by an Anomaly-based Behavior Analysis (ABA) method, with a baseline model to describe the normal behaviors of a system and detect malicious behaviors as those that deviate from the baseline. It describes the impact levels of different behaviors, such as turning on/off lights as low impact and opening the front door as high. It proposed a contextual array formation to join user identification, time slot, behavior sequence, behavior pair set, physical location, and gateway availability in the ABA. Although presenting promissory results, it needs a long time for training and depends on cloud computing.

Table 3.3: Intrusion Detection System works in smart homes

Work	Approach	Contributions	Limitations
Aegis: A Context-aware Security Framework for Smart Home Systems (Sikder et al., 2019a)	IDS based on context and behavior	Sensor-device co-dependence; context-awareness with proposed context array; Markov Chain model to detect anomalies; adaptive training	Dependence on cloud computing; dependence on historical data
Context Aware Anomaly Behavior Analysis for Smart Home Systems (Pan et al., 2019)	IDS based on context and behavior	Contextual array with diverse features; anomaly detection with ternary classifier	Dependence on cloud computing; dependence on historical data
A Supervised Intrusion Detection System for Smart Home IoT Devices (Anthi et al., 2019)	IDS based on behavior	Three layer ML with high accuracy for device profiling, detect wireless attacks and distinguish attack type	Dependence on historical and labeled data; does not consider the physical interface between users and devices
A Context-aware Framework for Detecting Sensor-based Threats on Smart Devices (Sikder et al., 2020)	IDS based on behavior	Comparison of different approaches for detecting sensor-based anomalies; real-time with low overhead	Depends on training time to be ready; focused on sensor-based threats

Anthi et al. (2019) describes a three-layer IDS based on supervised ML trained with network features. The three primary functions of the classifier are: to classify the type and profile the expected behavior of each IoT device connected to the network; identify malicious packets on the network when an attack occurs; and classify the type of the attack that has been deployed. The feature selection converted the PCAP files containing the network packets to a Packet Description Markup Language (PDML) format (Wireshark, 2021) including information about physical, data link, network, and transport layers with additional frame information and whether the data packet was inbound or outbound to an IoT device. It accurately detected Denial of Service (DoS), Man-in-the-Middle (MITM), scanning, and multi-stage attacks. The algorithm with the best classification results is the J48, a decision tree model. Furthermore, IP and TCP flags are the essential features. It is a generic implementation for IoT networks and can detect which device was affected by which attack. However, it requires previously labeled data in significant amounts to train the model and does not reckon the physical interface between users and devices.

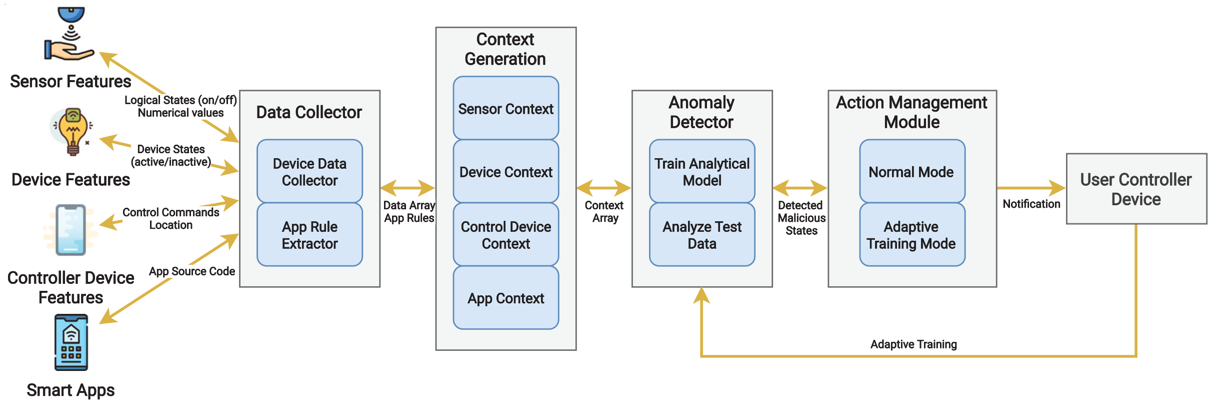


Figure 3.3: AEGIS architecture (Sikder et al., 2019a)

Sikder et al. (2020) presents 6thSense, a context-aware IDS based on sensor changes in smart devices to detect threats against sensor vulnerabilities. It carries the contributions from Sikder et al. (2019a), including sensor-device co-dependence. The goal is to detect malicious anomalies in smart devices, such as smartphones and smartwatches, by monitoring their sensor data, such as accelerometer, gyroscope, light sensor, proximity sensor, microphone, speaker, and camera. The adversary model considered three threats: triggering a malicious app via a sensor; information leakage via a sensor; and stealing information via a sensor. The evaluation investigated detection techniques based on Markov Chain, Naive Bayes, and other ML. Among the other ML techniques, the work employed PART as the rule learning method, logistic regression as the regression method, Multilayer Perceptron (MLP) as the neural network method, and J48, Logistic Model Tree (LMT), and Hoeffding tree as the decision tree methods. The LMT reached the best metrics results compared to the other ML-based techniques. Thus, comparing the Markov Chain, Naive Bayes, and LMT models, the Markov Chain and LMT have the best performance results. All three methods presented low overhead on CPU, RAM, disc, and power usage. 6thSense consists of the first comprehensive context-aware security solution against sensor-based threats. However, it depends on training time to be ready and focuses on sensor-based threats, thus not suitable for smart home security from a holistic perspective.

3.4 CONTINUOUS AUTHENTICATION

This section addresses the security works in smart homes related to continuous authentication. VAuth, presented in Feng et al. (2017), is a wearable security system to provide virtual assistants with an additional security channel based on voice physics. This approach minimizes impersonation attacks with good usability for users. People use eyeglasses, earphones, or necklaces with embedded VAuth that collects body-surface vibrations from the user and matches with received speech signals from the voice assistants. This proposed mechanism guarantees that the voice assistants execute only voice-owner commands. It endures against replay, mangled voice, or impersonation attacks with low energy and latency overheads. The vulnerability of this work remains in the reliability of a single source of truth, the proposed token. An attacker can explore it, considering it would depend on the token functionality. The collected data to perform continuous authentication is less prone to falsification if it originates from multiple sources.

Ashibani et al. (2019) proposes a holistic method to device security in a smart home, as shown in Figure 3.4. The continuous authentication takes into account context information from multiple sources. Furthermore, this work elaborated a taxonomy of contextual information regarding its transformations, gathering, and quality. The information collected from the environment can be classified as direct, which means the final data is achieved without additional operations, like activities, profiles, or families. The collected information can be indirect, which refers to those achieved by performing operations on contextual values, like calculating power consumption using voltage or speed using GPS locations. It also organizes the context according to its dynamism (changes in frequency) or locality (internal or external related to the system). It discusses two popular approaches for measuring the Quality of Context Information (QoC). It includes statistical analysis based on mathematical models, and confidence values utilizing different confidence values for devices and authentication tools. The components of the system are home devices (IP camera, thermostat, smart lock), end-user devices (smartphone, tablet), and home gateway (local server). The typical workflow of new users comprehends registration, verification (homeowner review), login, and usage.

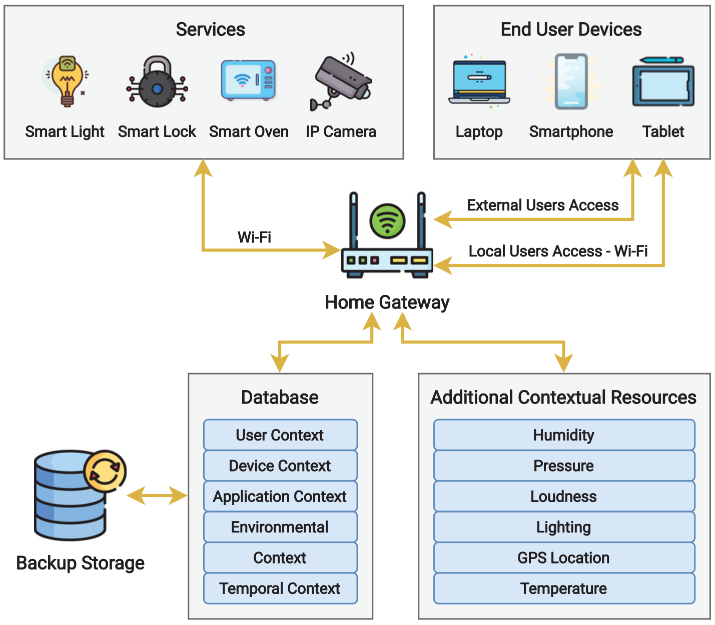


Figure 3.4: Proposed architecture in (Ashibani et al., 2019)

The framework copes with the addition and removal of new devices and users. The evaluation of the proposed system included IP address-based location, Bluetooth-based location, static credentials (username, password), and Google calendar data. The results demonstrated low latency overhead, effects on access decision-making by authentication-assigned weights and thresholds, and the capacity to handle multiple simultaneous requests without bottlenecking access to smart devices. One of the drawbacks consists of dependency on external information (Google Calendar), which may decay reliability in the system because an attacker can alter or suppress this data. Moreover, it remains dependent on end-user devices (smartphone, tablet), considering the interaction inside a smart home can happen using other ways, like voice assistants and direct contact. It decreases the likelihood of correct usage from users by forcing them to use only the smartphone, making the smart home vulnerable to misuse. Besides, relying on the smartphone can lead to violations since it has a general purpose and is carried in external environments for work and leisure, being more exposed to exploitation.

Table 3.4: Continuous Authentication works in smart homes

Work	Approach	Contributions	Limitations
Continuous Authentication for Voice Assistants (Feng et al., 2017)	CA based on voice for virtual assistants	Wearable security token; additional channel that provides physical confirmation; minimize voice impersonation; easy usability	Dependence on a single data source; Dependence on a single access way (voice assistant)
Design and Implementation of a Contextual-Based Continuous Authentication Framework for Smart Homes (Ashibani et al., 2019)	CA based on contextual information	Taxonomy of Contextual Information; instant contextual factors for authentication; does not require constant user interactions	Dependence on external data (Google Calendar); dependence on a single access way (smartphone)
User Authentication for Smart Home Networks based on Mobile Apps Usage (Ashibani and Mahmoud, 2019)	CA based on behavior	Application access analysis authentication model; implicit analysis, no user interruption required	Dependence on a single access way (smartphone); dependence on historical data
Implicit and Continuous Authentication of Smart Home Users (Amraoui et al., 2020)	CA based on behavior	User Behavior Analytics (UBA); Anomaly Detection (AD); implicit analysis, no user interruption required	Dependence on cloud computing; dependence on historical data

Ashibani and Mahmoud (2019) details a behavior-based continuous authentication model for smart home networks. The goal is to identify users using their smartphone app to ensure the person trying to control the smart home is authentic. The work uses Machine Learning (ML) for access decisions. It collects the data from the usage, selects features in the data preprocessing phase, balances classes, trains the classifier continuously, and finally, grants or not the user access. This option is suitable for continuous authentication because the smartphone continually provides app access patterns. However, as discussed before, users cannot depend on a smartphone to control their smart home, and this work is highly dependent on that. Moreover, it depends on tracking app access logs creating historical data that can be vulnerable to an attacker, thus exposing data privacy. Amraoui et al. (2020) proposes another behavior-based continuous authentication model for controlling smart homes. This work focused on making implicit re-authentication without explicit user interaction (entering a password). It leverages User Behavior Analytics (UBA) and Anomaly Detection (AD) paradigms to implement the CA mechanism. UBA assumes users follow frequent patterns while using computing devices and resources and is mainly enabled by ML techniques. AD's idea remains to build a baseline model over standard data and detect deviations. The architecture relies on a cloud backend that aims to build normal user behavioral patterns and an online process that analyses commands requested locally in a hub or remotely in the cloud backend. The dependency on a high amount of data to achieve reasonable accuracy is a drawback because it exposes privacy by collecting and storing user behavior. Attackers could exploit the dependence on cloud computing by isolating the home from the Internet. Furthermore, it does not support different levels of security for users or devices, compromising service differentiation and device isolation. Thus an impersonation attack would gain unrestricted access regardless of the exploited user.

3.5 DISCUSSION

The most appropriate approach to ensure user authenticity while interacting with the smart home. Prior works focused against insider attacks rely on continuous context-based or behavior-based authentication with historical data and depend on cloud computing. Furthermore, they ignore other access ways commonly used nowadays, like voice assistants, house devices, or the device itself. A robust solution would be a hybrid context-aware behavior-based continuous authentication to prevent and mitigate impersonation attacks, considering all access ways to ensure security enforcement and using edge computing to guarantee low latency. According to Google (2021), acceptable latency for users interacting with Google Assistant is ideal when less than 200 ms, OK between 2 s and 5 s, and not acceptable if higher than 5s. The security system should not depend on an external network, be restricted to the local network, and minimize message exchange by storing device states as they interact with the system, thus being less exposed to interceptions and preventing privacy violations. It should also count on different security layers, with some devices and actions restricted to specific user roles. The capacity to add and remove devices without compromising the correct functionality of the security system is also an important feature. We analyze comparatively in Table 3.5 three of the works cited above and expose the main requirements of smart homes:

- Ensure the data and physical privacy of the inhabitants (privacy perception);
- Control access to devices in real-time (low latency);
- Process the data produced in-house and in real-time (spatial and temporal locality);
- Support addition and removal of devices over time (device extensibility);
- Prevent, detect and mitigate impersonation attacks (security against impersonation);
- Isolate resources by establishing appropriate access privilege levels to devices (device isolation);
- Support digital and physical user-device interaction (security enforcement).

Table 3.5: Comparative works against impersonation attacks in SHS

Requirements	**Comparative Works**		
	Ghosh et al. (2019)	**Sikder et al. (2019a)**	**Ashibani et al. (2019)**
Privacy perception (PP)	(-) system usage data in database with sensitive personal data (+) Auth based on context and behavior	(+) no sensitive personal data stored (+) IDS based on context and behavior	(-) system usage data in database with sensitive personal data (+) CA based on context
Low latency in response (LL)	no data on system latency	(+) ≈ 519 ms for 24 devices in adaptive mode (+) ≈ 210 ms for notification of malicious activity	(+) ≤ 98 ms for Internet access time (+) ≤ 20 ms for local access time
Spatial and temporal locality (STL)	(-) dependency on cloud computing (+) data from the local network	(-) dependency on cloud computing (+) data from the local network	(-) data external to the local network (Google Calendar) (+) data from the local network (+) data processed with edge computing
Device extensibility (DE)	(+) supports inclusion and exclusion of devices	(+) supports inclusion and exclusion of devices	(+) supports inclusion and exclusion of devices
Security against impersonation (SAI)	(+) different data sources for authorization (+) verifies user behavior (+) verifies context trust	(+) different data sources for authorization (+) verifies user behavior (+) verifies context trust	(+) different data sources for authorization (+) verifies context trust
Device isolation (DI)	(+) different levels of belief for actions with greater impact	(-) no different treatment for devices or users	(+) different security levels to protect most important services (+) configurable weights for data sources
Security enforcement (SE)	(-) requires user-device interaction with end-user devices	(-) requires user-device interaction with end-user devices	(-) requires user-device interaction with end-user devices

We organized the security systems state-of-the-art for SHS in Table 3.6 with the characteristics detailed in the previous sections and requirements of the works, classified in high or low regarding the adherence to the raised properties. The classifications of the comparative works from Table 3.5 are explained, but the others were also classified. Rahmati et al. (2018) is classified with high PP, as it does not use sensitive personal data; not informed LL; low STL, as it depends on cloud computing; high RE, as it supports device addition and removal; low SAI, as it does not present methods to avoid insider attacks; high RI, as it has different risk levels, isolating some critical devices; low SE, as it depends on the SmartThings platform to enforce the security system. Sikder et al. (2019a) is assorted with low PP, as it uses sensitive personal data; high LL, since the latency for 30 policies in a hard conflict took 1.21 s, in a soft conflict 0.73 s, and in a restriction policy 0.25 s; low STL because it depends on cloud computing; high RE, as it supports device addition and removal; low SAI, as it does not verify user authenticity; high RI, as the policies isolate critical devices and use priority lists for the users; low SE, as it depends on end-user devices to enforce the security system.

Quintal et al. (2020) is categorized with high PP because it does not use sensitive personal data; not informed LL; low STL, as it depends on cloud computing; high RE, as it supports resource addition and removal; high SAI, because it evaluates the trust based on user behavior; high RI, as it demands higher trust values for critical resources; high SE, as the security system is enforced in the document's usage. Dutta et al. (2020) is classed in low PP, as it uses sensitive personal data; not informed LL; low STL, once it depends on cloud computing; high RE, as it supports device addition and removal; high SAI, because it verifies user behavior and context; high RI, as it permits granular configuration with ontologies to isolate devices in specific occasions; low SE, as it depends on digital means to enforce the security system. Pan et al. (2019) is classified in high PP, as it does not use sensitive personal data; not informed LL; high STL, as it uses locally generated data processed with edge computing; high RE, as it supports device addition and removal; high SAI, because it verifies user behavior and context; low RI, because it does not differentiate critical devices or services; low SE, as it depends on end-user devices to enforce the security system. Anthi et al. (2019) is assorted with high PP, as it does not use sensitive personal data; high LL, because although taking 41 s in the training phase, it takes 0.4 s to detect attacks and 0.2 s to identify the attack; high STL, since it uses locally generated data and processed at the edge; high RE, as it supports device addition and removal; low SAI, as it does not verify user authenticity; low RI, as it does not differentiate critical devices or services; low SE, as it depends on digital means to enforce system security.

Sikder et al. (2020) is ranked with high PP, as it does not use sensitive personal data; not informed LL; high STL, since it uses locally generated data processed at the edge; high RE, as it supports the addition or removal of sensors; high SAI, as it monitors user behavior; low RI, as it does not differentiate the sensors; high SE, as the security system is enforced in the usage of the smart devices. Feng et al. (2017) is arranged with high PP, as it does not use sensitive personal data; high LL, as for successful matches it takes 300-830 ms, with an average of 364 ms, for unsuccessful matches it takes 300-830 ms, with an average of 319 ms, and still takes less than 1 s for commands with more than 30 words; low STL, as it depends on cloud computing; high RE, as it supports resource addition and removal; high SAI, as it monitors biometric feature; not applicable RI; high SE, as the security system is enforced in the voice usage. Ashibani and Mahmoud (2019) is categorized with low PP, as it uses sensitive personal data; not informed LL; high STL, since it uses locally generated data processed at the edge; high RE, as it supports resource addition and removal; high SAI, as it verifies user behavior; low RI, as it does not differentiate the apps; high SE, as the security system is enforced in the app's usage. Amraoui et al. (2020) is classed with high PP, as it does not use sensitive personal data; not informed LL; low STL, as it depends on cloud computing; high RE, as it supports device addition and removal; high SAI, because it verifies user behavior; low RI, considering it does not differentiate critical devices or services; low SE, as it depends on end-user devices to enforce system security.

Table 3.6: Works against impersonation attacks in SHS

Work	Characteristics						Requirements						
	Approach	Data	Model	Data Type	Computing	Access Way	PP	LL	STL	RE	SAI	RI	SE
Rahmati et al. (2018)	Auth	Risk	Permission	Instant	Cloud	Digital	High	Not Informed	Low	High	Low	High	Low
Ghosh et al. (2019)	Auth	Behavior & Context	Weighted Linear Regression	Historical	Cloud	End user devices	Low	Not Informed	Low	High	High	High	Low
Sikder et al. (2019b)	Auth	Context	ABAC	Instant	Cloud	End user devices	Low	High (1.21 s)	Low	High	Low	High	Low
Quintal et al. (2020)	Auth	Behavior & Context	Adaptive Ensemble Supervised Learning	Historical	Cloud	End user devices	High	Not Informed	Low	High	High	High	High
Dutta et al. (2020)	Auth & IDS	Behavior & Context	ABAC & Hidden Markov Chain	Historical	Cloud	Digital	Low	Not Informed	Low	High	High	High	Low
Sikder et al. (2019a)	IDS	Behavior & Context	Markov Chain	Historical	Cloud	End user devices	High	High (519 ms)	Low	High	High	Low	Low
Pan et al. (2019)	IDS	Behavior & Context	Anomaly Behavior Analysis	Historical	Edge	End user devices	High	Not Informed	High	High	High	Low	Low
Anthi et al. (2019)	IDS	Behavior	Supervised Machine Learning	Historical	Edge	Digital	High	High (0.6 s)	High	High	Low	Low	Low
Sikder et al. (2020)	IDS	Behavior	Machine Learning	Historical	Edge	End user devices	High	Not Informed	High	N/A	High	Low	High
Feng et al. (2017)	CA	Biometric	Support Vector Machine (SVM)	Historical	Cloud	Digital	High	High (830 ms)	Low	High	High	N/A	High
Ashibani et al. (2019)	CA	Context	Context Trust	Instant	Edge	End user devices	Low	High (98 ms)	Low	High	High	High	Low
Ashibani and Mahmoud (2019)	CA	Behavior	Machine Learning	Historical	Cloud	Smartphone	Low	Not Informed	High	High	High	Low	Low
Amraoui et al. (2020)	CA	Behavior	One Class Support Vector Machine (OCSVM)	Historical	Cloud	End user devices	Low	Not Informed	Low	High	High	Low	Low

4 THE ZASH SYSTEM

This chapter presents a zero-trust access control with context-aware and behavior-based continuous authentication for smart homes. The ZASH (Zero-Aware Smart Home) system runs on the network's edge and controls users' access to smart devices. Initially, Section 4.1 describes the system overview with characteristics, entities, and flow. Section 4.2 presents ZASH architecture and details its components. Lastly, Section 4.3 details the operation and algorithms.

4.1 SYSTEM MODEL

This section presents an overview of the assertions, environmental characteristics, the concerns of the users, and ZASH entities and flow. Smart homes count on infrastructure networks enabled by Ethernet, Wi-Fi, Bluetooth, Zigbee, etc.; the connection between devices is end-to-end with TCP and IP protocols. Despite the smart home trend becoming popular in recent years, people are still highly concerned about their privacy related to data and the physical world. SHS is not as secure as it should be regarding invaders, especially during impersonation attacks. Current works struggle to prevent, detect and mitigate these attacks due to the high complexity of the environment and the interaction between users and devices. Thus, this is one of the issues avoiding a broader adoption of smart homes. The objective of this work is to focus on the prevention and mitigation of impersonation attacks in an SHS.

In order to support ZASH's model operation, we suppose assertions about the smart home environment, the applied communication network, and the access control behavior. Regarding the SHS environment, ZASH takes into account the existence of passive and active smart devices collecting data from the user's behavior; the system always collects user interactions with smart devices; and these devices are never tampered with malicious behavior. For example, an active device is a door that needs direct user interaction to change its state. For instance, a passive device is a pressure sensor installed on the floor that detects the user's movements without them explicitly activating the sensor. A change of state on an active device is considered a user request, and a change of state on a passive device is regarded as a user interaction with the environment.

Concerning the communication network, the devices always communicate with the server and vice-versa without issues; and the Virtual Local Area Network (VLAN) where the system works is free of malicious agents. ZASH communicates only inside a VLAN protected by a firewall, which includes the smart devices and the local server and is not connected to the Internet. It can only be configured by a device inside VLAN with recently provided proof of identity from a root user. Lastly, in respect of access control, exists at least one administrator on the system capable of configuring it, and their access cannot be compromised; authentic users are always able to provide valid proof of identity; impersonated users are never able to provide valid proof of identity; the system can differentiate which user is interacting with the smart device; and the verification for proof of identity is always correct.

A SHS typically embodies several heterogeneous devices, denoted as $D = \{d_1, \dots, d_n\}$, where n is the number of devices. A VLAN connects the devices from SHS and can be a smart object, like a smart TV; an actuator, like a smart curtain; or just a sensor, like for temperature. ZASH specifically will count on User Levels (UL), denoted as $UL = \{UL_1, \dots, UL_m\}$ (e.g., visitor, child, adult, admin — from the least to the most privileged), where m is the quantity of ULs. The Device Classes (DC), denoted as $DC = \{DC_1, \dots, DC_k\}$ (e.g., non-critical and critical — from the least to the most important), where k is the quantity of DCs. Furthermore, there are Actions (A) that a user can perform on a device, denoted as $A = \{A_1, \dots, A_p\}$ (e.g., view, control, manage — from the least to the most impactful), where p is the quantity of As. The ZASH system imposes barriers to attackers exploiting stolen credentials by counting on different UL, DC, and A to guarantee the isolation of devices and differentiation of actions. It has a root user, the only access to the local server to configure the model parameters. As this user is not the same as admin UL, it does not participate in the ZASH authorization, only being employed to log in to the dedicated machine. The root can manage devices by adding new, altering, and deleting existing ones, which must be classified with a DC. Hence, this implies on its Security Level (s), a numeric value representing the required level to perform an action on a device with a user level and is determined as $s \in \mathbb{N} \mid s \leq 100$. Similarly, the root manages users by adding new, altering, deleting existing ones, and assigning them an UL.

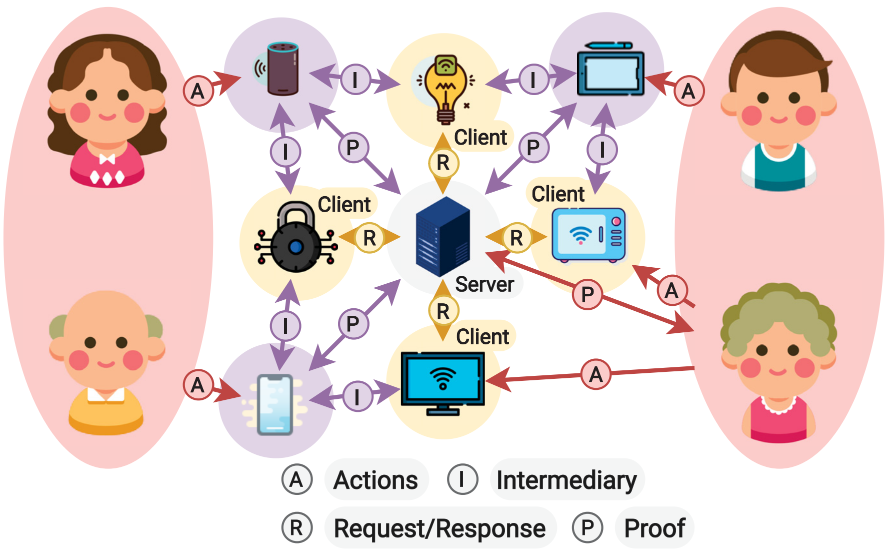


Figure 4.1: The ZASH flow

ZASH comprises the client side in the smart devices and the server side in a local machine. The ZASH flow, detailed in Figure 4.1, consists of the following entities: *users*, which represents the inhabitants and visitors; *intermediaries*, which the users may use to access the smart home; *smart devices*, which are the smart objects, actuators, and sensors; and *server*, localized in a local machine at the edge network. The users perform actions using intermediaries (e.g., smartphone, tablet, voice assistant) or interacting directly with a device to control its states. The smart devices, like a smart light, receive the user request and have the ZASH client-side that redirects to the local server-side that authorizes it or not. ZASH can also require proof of identity from users in suspicious requests, hence granting or denying it. The decision flows back to intermediaries (if used) and finally for devices to execute or not an action. Whether the user interacts directly with the smart device without any intermediary, ZASH uses the intermediary closer to the user to collect the identity proof.

The ZASH communication protocol, as shown in Figure 4.2, comprehends the messages exchanged by the local server and smart devices and shows their fields. The request message carries the server IP address, the message destination; the device id, identifying the source of the message; the user id, meaning the user who interacted with the device; the context, encompassing contextual factors associated with the device, such as access way; and the action, the user's requested action on the device. The response message holds the device IP address, the message destination; the authorized value, represented by a boolean indicating the outcome of the requested action; and the original action. When the request fails to meet the ZASH authorization requirements, it is flagged as suspicious and triggers the generation of two additional messages. The server sends a request for user identity proof to the intermediary device of the user, and the intermediary device responds with a proof response message to the server.

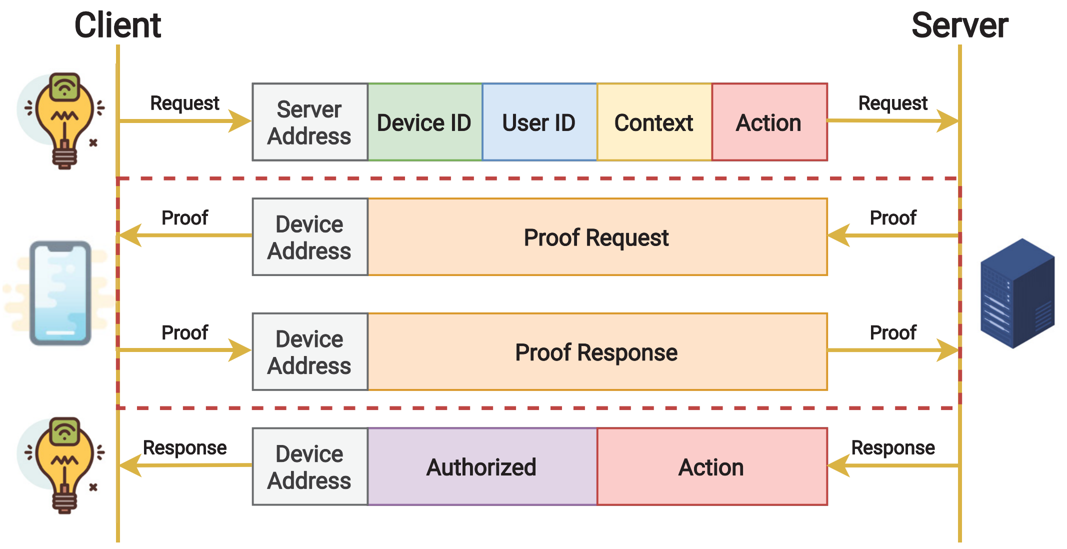


Figure 4.2: Protocol flow with request and response messages

For a better understanding, we described the server and client state machines in Figure 4.3 and Figure 4.4. The server starts and listens for new connection requests from the devices, which starts and requests a TCP connection with the server. They complete the three-way handshake and establish a reliable connection open to exchange messages. The devices will request authorization upon user interaction by sending a message to the server, which will authorize it or not. The connection should always remain open and only close when the device requests to stop operating. The server should never shut down, as the system would stop working.

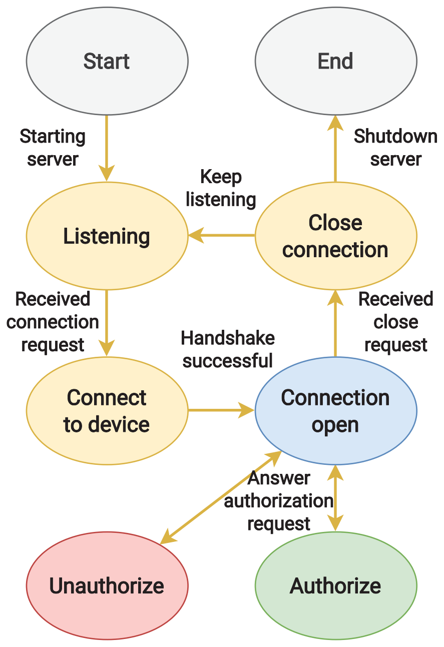


Figure 4.3: Server state machine

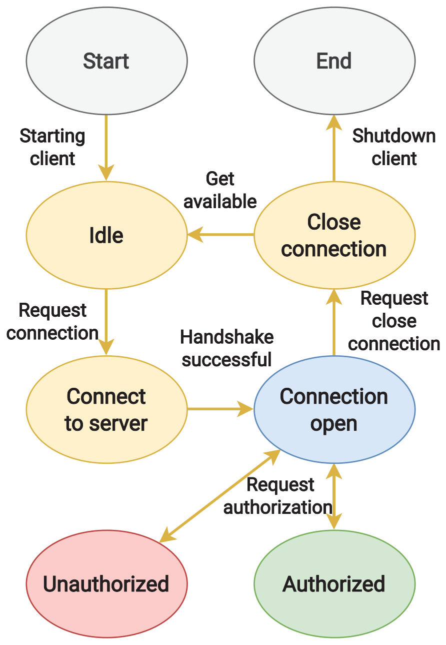


Figure 4.4: Client state machine

4.2 ARCHITECTURE

The ZASH architecture, as shown in Figure 4.5, contains the client-side, composed only of the Device Enforcer, representing the Policy Enforcement Point (PEP) that redirects the request for the server-side that performs the access control. The server-side embraces three modules, named **Behavior, Collection** and **Decision**. The Behavior module includes *Configuration Assembler*, where the root user configures the system behavior representing the Policy Storage, and *Notification Dispatcher*, responsible for notifying users to trigger further actions from them. The Collection module comprises the *Device Communicator*, which involves all devices configuration from the SHS and is responsible for communicating with them, and *Data Provider*, which stores the latest device states and processes them to support decisions. Finally, the core Decision module, representing the Policy Decision Point (PDP), consists of the *Activity Manager*, which process activities in the SHS, the *Context Manager*, responsible for building instant context information, the *Ontology Manager*, which comprehends the rules for the authorization, and the *Authorization Controller*, in charge of using all provided information to accept or reject a request. The Ontology Manager, Context Manager, and Activity Manager represent the Policy Information Point (PIP), while the Authorization Controller represents the Zero Trust Engine (ZTE). The system regards any request reaching the Device Enforcer as untrustworthy and depends on the Authorization Controller to assume it is trustworthy. ZASH does not execute the other verifications if the request fails in the Ontology Manager. Further, in case it fails in the Context Manager, the Activity Manager is not verified.

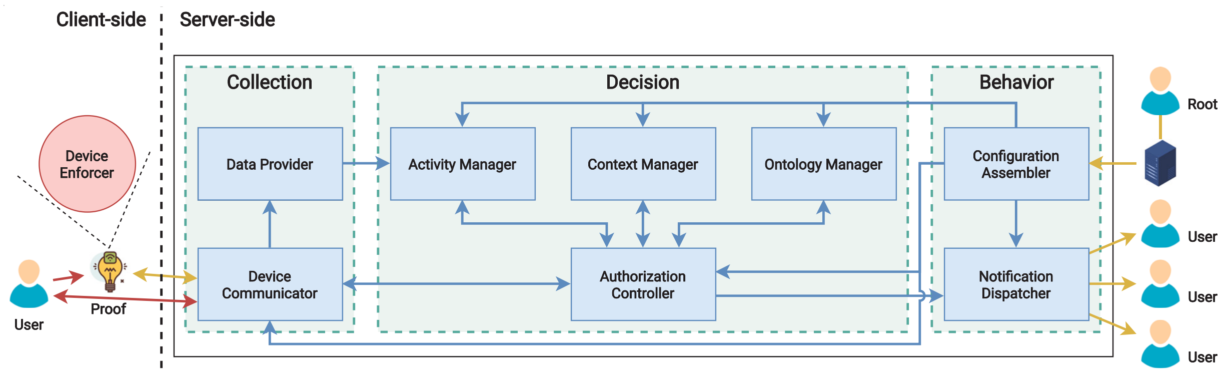


Figure 4.5: The ZASH architecture

4.2.1 Device Enforcer

The **Device Enforcer** is the client side of ZASH and is responsible for enforcing the access control in smart devices by redirecting the decision to the local server. After the initial setup, ZASH is ready to receive requests, automatically redirected from devices using any suitable technology, like Zigbee or Wi-Fi, to the local server. Algorithm 1 describes the steps followed by each device when users act on active devices or change the state in passive devices. The device forms *Request* information with the server IP address, device ID, user ID, context array, and action to send to the server (*l.*4-5), then it receives a response and acts if authorized (*l.*7-8). Considering any device state change passes through the local server, the system knows all the latest states and does not require pulling it from all other devices in every request, thus minimizing the data traffic.

Algorithm 1 Device awaits ZASH authorization

```
1: DeviceEnforcer
2: $ServerAddress$
3: procedure ListenStateChange                    ▷ Triggered when device detects state change
4:     $Request \leftarrow ServerAddress, Device, User, Context, Action$
5:     send $Request$                                    ▷ Sends request for authorization
6: procedure ListenAnswer($Response$)
7:     if $Response.Authorized$ is $True$ then
8:         perform $Response.Action$                     ▷ Perform action if authorized
```

4.2.2 Behavior Module

This module defines the system behavior with the **Configuration Assembler** component while triggering further actions from users with the **Notification Dispatcher** component. Hence, ZASH starts operating after the initial configuration by the root using Configuration Assembler, as shown in Algorithm 2, which assigns UL for each user (l.3), DC for each device (l.3), and ontologies rules (l.3). Root also configures s for each C, DC, UL and A (l.3), the number of rejected requests allowed by the user before blocking (l.4) and the considered time interval (l.4). The build interval (l.4) is also configured by the root to tell ZASH when the building of the Markov Chain and other probabilities are reliable to start operating. Finally, the activity threshold (l.4) defines the minimal probability of considering an activity as valid. As it can be cumbersome, the system should come with at least two predefined profiles: hard, with more strict rules as default, and soft, with less strict parameters. The Notification Dispatcher sends alerts (l.7-8) when requested.

Algorithm 2 Configuration of parameters and dispatch of alerts

```
1: ConfigurationAssembler
2: procedure SetAllConfiguration($U, D, O, SL, BT, BI, BdI, AT$)
3:     $Users \leftarrow U; Devices \leftarrow D; Ontologies \leftarrow O; SecurityLevels \leftarrow SL;$
4:     $BlockThreshold \leftarrow BT; BlockInterval \leftarrow BI; BuildInterval \leftarrow BdI; ActivityThreshold \leftarrow AT$
5: NotificationDispatcher
6: procedure AlertAllUsers($BlockedUser$)
7:     for $User$ in $Users$ do
8:         $SendMessage(BlockedUser, User)$              ▷ Dispatch alert messages to users
```

4.2.3 Collection Module

This module collects information from the smart devices in the **Device Communicator** component and stores the latest states of the smart devices in the **Data Provider** component. Furthermore, the system assumes that any user can interact with any device in the SHS in any way (e.g., directly, smartphone, tablet), but the requested action will be authorized or not by ZASH. The Data Provider only stores the latest device states, as shown in Algorithm 3. Algorithm 4 describes that Device Communicator is responsible for managing proof of identity by collecting from the user when needed (l.5), storing the proofs (l.9), and clearing those over T time (l.12). The proof is stored for the user and access way for T time, so the user is not bothered to prove identity every time using the same access way, improving the user experience. Furthermore, ZASH receives a request from a device, where Device Communicator updates the current device state in Data Provider (l.14) whether the action is *control* (l.13) and then asks Authorization Controller (l.17) for active devices performing control action or any device executing view or manage actions (l.16). Then, it sets the current state as the last state if the change was granted (l.21) and removes (*l*.23) or adds (*l*.25) a device whether the action is *manage*. Finally, it returns the *Response* to the device to execute or ignore the requested action (*l*.27).

Algorithm 3 All latest devices states are stored

```
1: DataProvider
2: procedure UpdateCurrentState($Request$)
3:     $CurrentState \leftarrow Copy(LastState)$
4:     $CurrentState_{Request.Device.Id} \leftarrow InvertState(CurrentState_{Request.Device.Id})$
5: procedure UpdateLastState
6:     $LastState \leftarrow CurrentState$
```

Algorithm 4 Device communication and identity proofs management

```
1: DeviceCommunicator
2: function ExplicitAuthentication($User$)
3:     $Proof \leftarrow FindProof(Request.User.Id, Request.Context.AccessWay)$    ▷ Find proof for user with access way
4:     if $Proof$ not found then
5:         $Proof = InputProof()$    ▷ Wait for user proof of identity
6:         if $Proof$ not matches $User$ then
7:             return $False$    ▷ Deny action
8:         else
9:             store $Proof$    ▷ Store proof to be used for the next T time
10:    return $True$    ▷ Grant action
11: function ListenRequest($Request, CurrentDate$)
12:    $ClearProofs(CurrentDate)$    ▷ Clear stored proofs obtained more than T time ago
13:    if $Request.Action$ is $Control$ then
14:        $DataProvider.UpdateCurrentState(Request)$    ▷ Update current state with requested action
15:    $Result \leftarrow True$
16:    if $Request.Device.IsActive$ or $Request.Action$ is $View$ or $Request.Action$ is $Manage$ then
17:        $Result \leftarrow AuthorizationControl.AuthorizeRequest(Request, CurrentDate,$
           $ExplicitAuthentication)$
18:    $Authorized \leftarrow False$
19:    if $Result$ is $True$ then
20:        $Authorized \leftarrow True$
21:        $DataProvider.UpdateLastState()$    ▷ Update last state to be former current state if granted
22:        if $Request.Action$ is $Manage$ and $Request.Device$ exists then
23:            $AuthorizationControl.ConfigurationAssembler.RemoveDevice(Request.Device)$
24:        else if $Request.Action$ is $Manage$ and $Request.Device$ not exists then
25:            $AuthorizationControl.ConfigurationAssembler.AddDevice(Request.Device)$
26:    $Response \leftarrow Request.Device.Address, Authorized, Request.Action$
27:    send $Response$    ▷ Send answer back to device
```

4.2.4 Decision Module

This module analyzes the user request to decide on access, composed of the Authorization Controller, which uses three components: ontologies verification, context trust evaluation, and activities assessment leveraging the access control robustness. Algorithm 5 details the **Authorization Controller** that, when more than a defined number of rejected requests occurs for the same user in a defined time interval, blocks the user and notifies all users in the system through their registered personal devices (*l*.10-12). Only the root user can unblock it through the local dedicated machine. The Authorization Controller verifies firstly with the Ontology Manager (*l*.6), which in turn checks for the rules set by the root user, as shown in Algorithm 6. Secondly, it verifies with the Context Manager (*l*.7) to certify the request has the expected context trust based on the combination of security level from $DC_s + A_s$ and $UL_s + A_s$, as shown in Algorithm 7. Finally, it feeds the Activity Manager (*l*.8) to build the Markov Chain transition matrix and then check if the change from the last state to the current state is above a threshold P, as shown in Algorithm 8. If the requested action fails in the Context Manager or the Activity Manager, the Device Communicator asks for proof of identity from the user to validate the action. A valid proof, a matching fingerprint or facial recognition from the user interacting with the device, permits the system to learn the correct action and to recalculate the time and state transition probabilities considering the new valid possibility.

Algorithm 5 Authorization control for user requesting action on device

```
1:  AuthorizationControl
2:  function AuthorizeRequest($Request, CurrentDate, ExplicitAuthentication$)   ▷ Triggered every request
3:      $ClearUsers(CurrentDate)$   ▷ Clear rejects out of interval from all users
4:      if $Request.User.IsBlocked$ then
5:          return $False$   ▷ Deny action
6:      if not $VerifyOntology(Request)$ or
7:          not $VerifyContext(Request, CurrentDate, ExplicitAuthentication)$ or
8:          not $VerifyActivity(Request, CurrentDate, ExplicitAuthentication)$ then
9:          $Request.User.Rejected \leftarrow Request.User.Rejected \cup \{CurrentDate\}$   ▷ Register rejection
10:         if $|Request.User.Rejected| > ConfigurationAssembler.BlockThreshold$ then
11:             $Request.User.IsBlocked \leftarrow True$   ▷ Block user if above threshold
12:             $AlertAllUsers(Request.User)$   ▷ Alert all users about blockage
13:         return $False$   ▷ Deny action
14:     return $True$   ▷ Grant action
```

The **Ontology Manager** intersects A on a DC from a specific UL to verify ontologies defined by root through the Configuration Assembler. Ontology-based modeling is the best technique to represent knowledge through formalisms (de Matos et al., 2017), providing flexible and customized configuration with the ability to represent complex structures compared to other approaches, such as key-value or object-based modeling. Figure 4.6 represents an instance for the rules of valid ontologies, where each UL_i for a specific DC_j can perform a set of A, named Capabilities (Cap_{ij}) (Equation 4.1), and UL_i inherits all Cap_{i-1j} from the lower UL_{i-1} (Equation 4.2). The Ontology Manager is mainly responsible for dealing with easily detectable attacks and keeping the system safe and secure, together with the Context Manager, while the Activity Manager is starting.

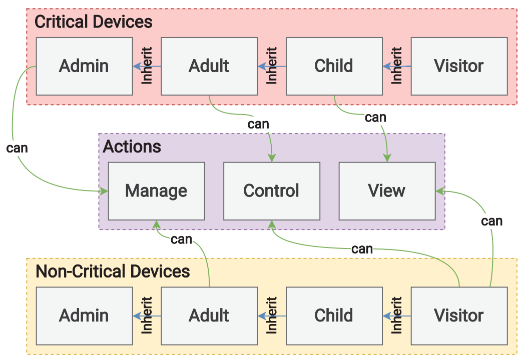


Figure 4.6: ZASH Ontologies

$$Cap_{ij} \subseteq A, \ \forall i = 1..4, \forall j = 1..2 \qquad (4.1)$$

$$Cap_i \supseteq Cap_{i-1}, \ \forall i = 2..4 \qquad (4.2)$$

Algorithm 6 Ontologies verification for access control

```
1: OntologyManager
2: function VerifyOntology($Request$)                                        ▹ Check ontologies for request
3:     $Cap \leftarrow FindCap(Request.User.UserLevel, Request.Device.DeviceClass)$    ▹ Find capabilities for UL in DC
4:     if $Request.Action$ in $Cap$ then
5:         return $True$                                                      ▹ Grant action
6:     else
7:         return $False$                                                     ▹ Deny action
```

The **Context Manager** works with Security Level (s), determined as $s \in \mathbb{N} \mid s \leq 100$. Root configures initial s of each DC and UL. They also set an additional s for each possible A. The Context Manager combines s of both DC (DC_s) with A (A_s) and UL (UL_s) with A (A_s) to check which one has the highest value. This value verifies whether instant context information achieves the required s. Instant context counts on factors, as instantiated in Equation 4.3 (e.g., request time, access way, localization, age, group). The system adds up and compares each of these context factors s. For example, the time can be classified as common or uncommon if the request time occurrence probability is below a threshold. The access way can be requested (the device itself), house (e.g., voice assistant, tablet), or personal (smartphone). The localization can be divided into internal or external about the house. The age can be linked to each user and can be adult, teen, or kid. The group can be ranked together or alone. Ashibani et al. (2019) utilized a similar technique to compare security level access to devices threshold with confidence level collected from context and demonstrated low-performance overhead, flexible access control, and high scalability. Equation 4.4 represents the sum (X_s) of the context factors (C), and Equation 4.5 expresses the subsequent logic to accept the request. Figure 4.7 illustrates the flow performed by the decision process of the Context Manager. ZASH prompts for proof of identity if the context trust is below the expected security level to assure user authenticity. This proof needs to be a *Something You Are* challenge, which is less prone to falsification and can be a fingerprint, face recognition, voice recognition, etc., as employed in Dimitrakos et al. (2020).

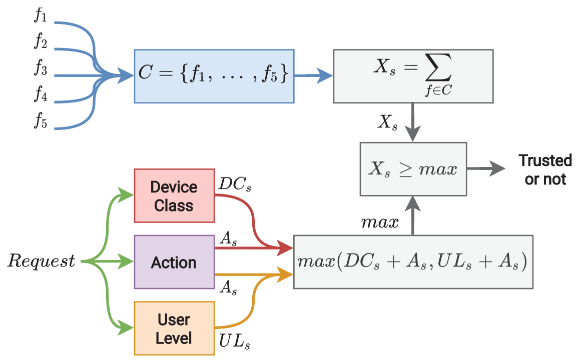


Figure 4.7: Context Manager Flow

$$C = \{f_1, \dots, f_5\} \mid f_i \leq 30, \forall i = 1..5 \tag{4.3}$$

$$X_s = \sum_{f \in C} f,\ X_s \in \mathbb{N} \mid X_s \leq 100 \tag{4.4}$$

$$X_s \geq max(\{DC_s + A_s,\ UL_s + A_s\}) \tag{4.5}$$

Algorithm 7 Context trust evaluation for access control

```
1:  ContextManager
2:  function VerifyContext($Request, CurrentDate, ExplicitAuthentication$)    ▷ Check context for request
3:      $CalculateTime(Request, CurrentDate)$    ▷ Calculate if request time is common or uncommon
4:      $CheckBuilding(CurrentDate)$    ▷ Check if time probability building is over
5:      if $IsTimeBuilding$ then
6:          $Request.Context.Time \leftarrow COMMOM$    ▷ Time is always the highest trust while still building
7:      $ExpectedDevice \leftarrow Request.Device.DeviceClass.SecurityLevel+$
        $Request.Action.SecurityLevel$    ▷ Calculate security level for A in DC
8:      $ExpectedUser \leftarrow Request.User.UserLevel.SecurityLevel+$
        $Request.Action.SecurityLevel$    ▷ Calculate security level for A in UL
9:      $CalculatedTrust \leftarrow CalculateTrust(Request.Context, Request.User)$    ▷ Calculate context security
    level
10:     if $min(CalculatedTrust, 100) < min(max(ExpectedDevice, ExpectedUser), 100)$ then
11:         if not $ExplicitAuthentication(Request.User)$ then    ▷ Require proof of identity
12:             return $False$    ▷ Deny action
13:     return $True$    ▷ Grant action
```

The **Activity Manager** counts on sequential activities that follow a specific pattern. Activity Manager relies on a Markov Chain fed with every successful user request, changing the transition matrix's balance over time. Each UL has its respective Markov Chain since they tend to present different patterns. It is associated with UL only to preserve individual users. The Activity Manager starts inactive in the authorization module as it processes some requests to create a reliable transition matrix over time. The other modules mitigate the cold start problem. Sikder et al. (2020) proved that Markov Chain is the best technique to detect anomaly requests in a context-aware environment, and Sikder et al. (2019a) applied the technique to achieve high accuracy in an SHS with low-performance overhead, making it suitable for edge computing. Each activity (Ay_i) carries information at that moment about every device state (DS_i) that is binary 0 or 1 data. Equation 4.6 represents Ay_i, where n is the number of devices in the SHS. Figure 4.8 illustrates the transition model, where m is the number of all possible states Ay (Equation 4.7) and P_{ij} is the probability of going to state j at time $t + 1$ from state i at time t as shown in Equation 4.8. Considering each DS_i is binary, the value of $m = 2^n$. Similarly to the Context Manager, ZASH requires proof of identity if the transition probability between two states is below a threshold.

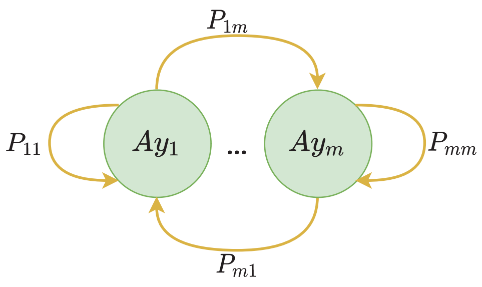


Figure 4.8: ZASH Markov Chain

$$Ay_i = \{DS_1, \dots, DS_n\},\ n \in \mathbb{N} \tag{4.6}$$

$$Ay = \{Ay_1, \dots, Ay_m\},\ m \in \mathbb{N} \tag{4.7}$$

$$P_{ij} = Pr(Ay_j \mid Ay_i),\ \forall 1 \le i, j \le m \tag{4.8}$$

Algorithm 8 Activities assessment for access control

```
1: ActivityManager
2: function VerifyActivity($Request, CurrentDate, ExplicitAuthentication$)    ▷ Check activities for request
3:     $CheckBuilding(CurrentDate)$    ▷ Check if Markov chain building is over
4:     $CurrentState \leftarrow DataProvider.CurrentState$
5:     LastState$\leftarrow$ $DataProvider.LastState$
6:     if not $IsMarkovBuilding$ then    ▷ Always pass if still building
7:         if $MarkovChain.GetProbability(CurrentState, LastState) < P$ then    ▷ Check if transition above threshold
8:             if not $ExplicitAuthentication(Request.User)$ then    ▷ Require proof of identity
9:                 return $False$    ▷ Deny action
10:    $MarkovChain.BuildTransition(CurrentState, LastState)$    ▷ Build the transition matrix
11:    return $True$    ▷ Grant action
```

4.3 OPERATION

We illustrate the ZASH operation in an SHS setting to highlight the functioning of each component. We present a simple network setting in a local dedicated machine for the comprehension facility. Figure 4.9 shows an initial configuration for ZASH, depicting the system operation in the following steps. Figure 4.10 represents the ontologies verification, where the Ontology Manager validates the request by User 1 to control Device 1. As User 1 has the admin user level, Device 1 has the non-critical device class, and the configured ontologies allow the admin user level to control a device with the non-critical device class, the Ontology Manager defines the request as valid.

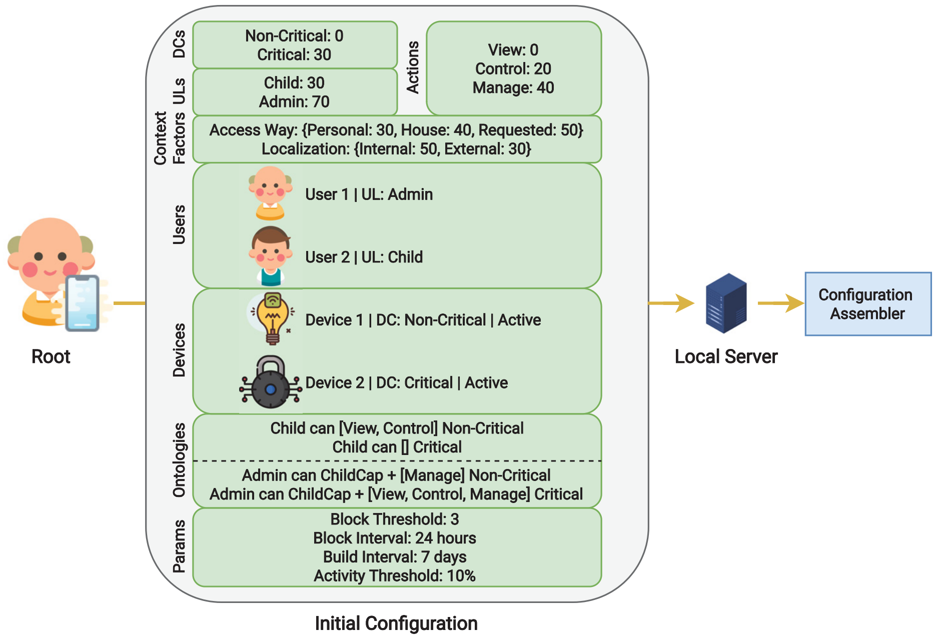


Figure 4.9: Example of initial configuration

The Context Manager component then verifies the request. In this case, as shown in Figure 4.11, it considers the context with the access way factor as *house* (e.g., tablet or smart assistant) with a trust value of 40, and the localization factor as *internal* with a trust value of 50. Thus, adding the two values results in the context trust of 90. Meanwhile, the *non-critical* device class has a security value of 0, the *control* action of 20, and the *admin* user level of 70.

Therefore, adding 0 to 20 results in 20 and 70 to 20 results in 90, the max security value is 90. As the context trust of 90 equals the security level of 90, the request is hence seen as trusted.

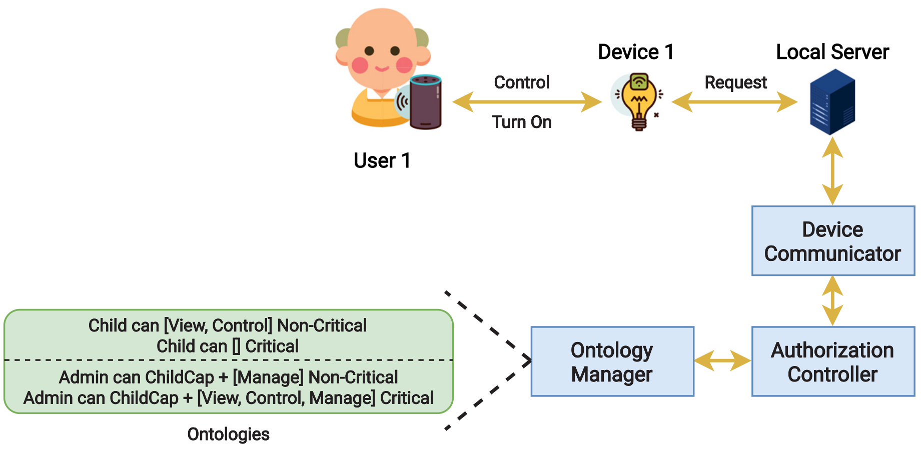


Figure 4.10: Example of the Ontology Manager operation

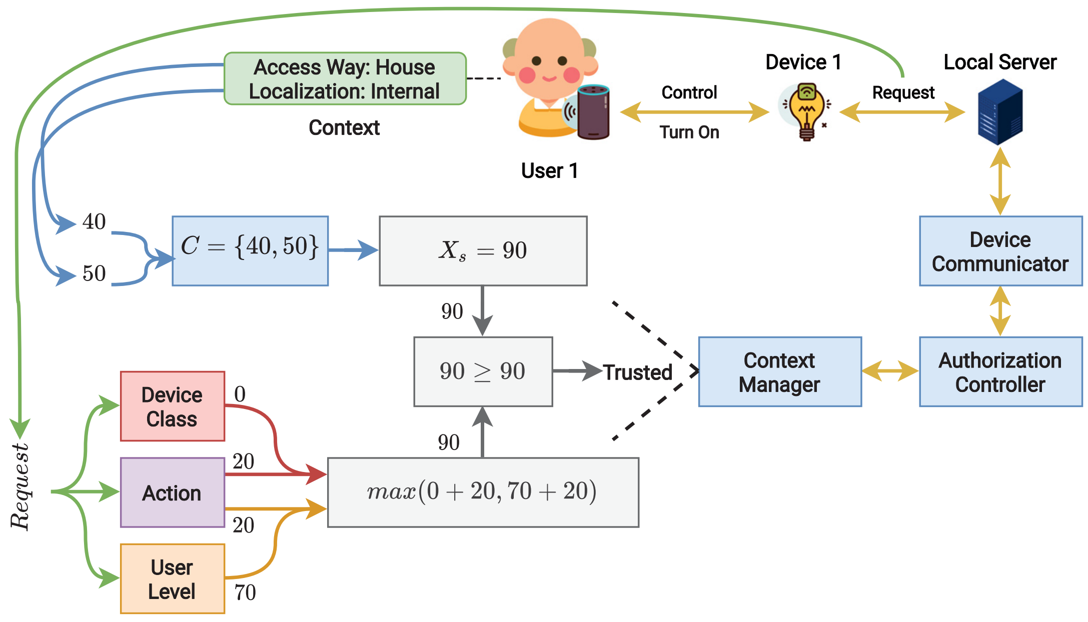


Figure 4.11: Example of the Context Manager operation

Lastly, the Activity Manager component also checks out the request. As shown in Figure 4.12, it takes the data provider's current and last states to recalculate the probability matrix. As this is the first request, the previous state is [0,0], meaning Device 1 and Device 2 are off, and the current state is [1,0], meaning Device 1 is on and Device 2 is off. Hence, the system creates the new state in the Markov Chain and calculates the probability as 100% from the [0,0] state to the [1,0] state, as shown in Section 4.3 because it is the only transition so far. Equation (4.10) represents the transition matrix after the first request. As the requests are processed, the Activity Manager increases the state space, recalculates the Markov Chain probabilities, and updates the transition matrix. Since the defined threshold to consider an activity suspicious is below 10%, and the transition probability is 100%, the action is rated normal. Hence, as all components verified the action and passed, the device was finally granted to perform.

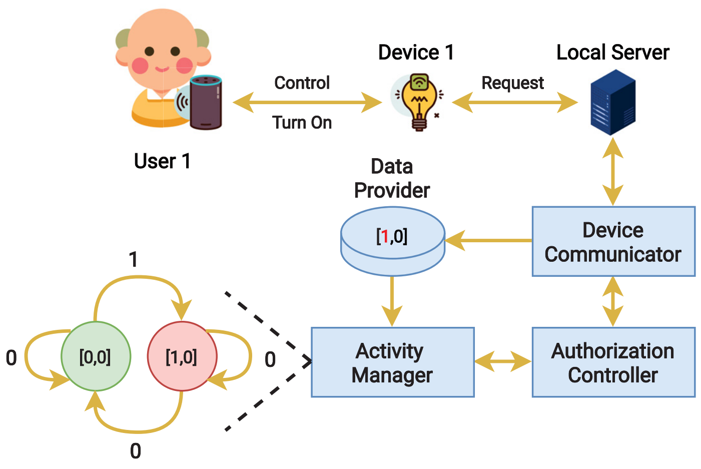


Figure 4.12: Example of the Activity Manager operation

$$P(X_{t+1} = [1,0] | X_t = [0,0]) = 1 \qquad (4.9)$$

$$P = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad (4.10)$$

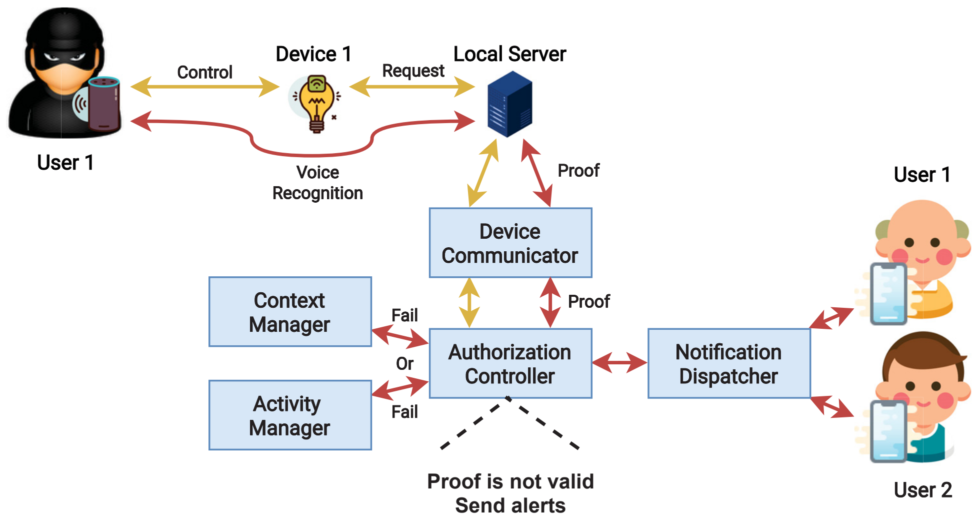


Figure 4.13: Example of invalid proof collection and sent alerts

Figure 4.13 shows an attacker trying to control Device 1. The attacker impersonates User 1 and then tries to control Device 1 through the voice assistant. Hence, the request fails in the Context Manager or Activity Manager. Further, ZASH employs voice recognition to ensure User 1 authenticity, but it does not match. In this way, the system sends an alert to the personal devices of User 1 and User 2, which can take further actions, such as calling the police. In case User 1 fails to prove their authenticity more than three times in 24 hours, as shown in Figure 4.14, on the fourth fail, the system blocks User 1, and consecutive requests are denied.

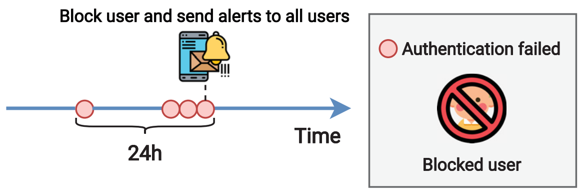


Figure 4.14: Example of the user blockage

4.4 SUMMARY

This chapter presented the ZASH system, detailing its flow, architecture, and operation. The flow consisted of the interaction between the entities, named users, intermediaries, smart devices, and ZASH. The user's importance in the flow is outlined as the reason for the system, considering we proposed ZASH to protect their privacy and security. The architecture reflects the requirements for privacy perception, low latency, spatial and temporal locality, device extensibility, security against impersonation, and device isolation. The decision process counts on three complementary layers (ontologies, context, and activities) to make access control more robust and less exploitation-prone. The algorithms displayed the code structure to support the theory and the mechanism to enforce access control. Finally, the operation exposed an example to illustrate the ZASH functionality.

5 EVALUATION

This chapter details the evaluation of ZASH's robustness, efficiency, extensibility, and performance. Section 5.1 presents the tools and methodology employed to evaluate ZASH. Section 5.2 describes the correctness test of ZASH's algorithms. Finally, Section 5.3 details results from a more robust evaluation performed in the ns-3 network simulator.

5.1 TOOLS AND METHODOLOGY

This section presents the defined methodology and tools to evaluate ZASH. We used for all the executions the following hardware: *Intel® Core™ i7-8565U CPU @ 1.80GHz × 8, NVIDIA GeForce MX150/PCIe/SSE2, 15,4 GiB RAM, and 64-bit OS type*. The operational system was *Zorin OS 15.3*. We adopted the methodology to validate the correctness and viability of the system in *Python*, then make a more robust evaluation using the *ns-3 network simulator*. For that, we implemented and executed the ZASH system in three approaches: the first in *Python (version 3.9.5)*; the second in pure *C++ (version 17)*, because the *ns-3* only works with *C++* for internal modules and to assure the migration from *Python* did not cause any issue; and finally the third in the *ns-3 (version 3.36.1)* structure.

We chose *Python* programming language for the *preliminary evaluation*[1] because of its portability, ease of file manipulation, and ease of algorithm writing, allowing rapid development and evaluation. Then we selected *C++* for the *second implementation*[2] to validate the code before inserting it in *ns-3*. We chose *ns-3* for the *third implementation*[3] because it is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. Furthermore, *ns-3* is free, open-source software, licensed under the GNU GPLv2 license, and maintained by a worldwide community (Consortium, 2021). It supports customized behavior with *C++* and *Python* codes, configurable topologies, and the usage of well-known communication technologies, like Wi-Fi and 6LoWPAN.

The dataset for all approaches was the *SIMADL (Simulated Activities of Daily Living Dataset)* (Alshammari et al., 2018)[4]. We chose this dataset because it simulates daily life activities and presents a good amount of data in a well-structured form with 29 devices, as shown in Table 5.1. We selected the *d6_2m_0tm.csv* variant (167,211 lines of records after cleaning duplicates, 60 days, 1 line per second) for presenting the biggest amount of records compared to the other variants of this dataset. Device states are represented with 0 or 1, being, for example, a door with state 0 closed and 1 opened. There are eight passive and 21 active devices in the dataset. The home design, exhibited in Figure 5.1, includes a bedroom, living room, kitchen, bathroom, office, and hallway. Alshammari et al. (2018) utilized this arrangement of a single-bedroom layout to create the dataset, representing a simplistic but realistic case. We configured all smart locks *critical* device class because an attacker could invade the house physically or even lock the user without consent by controlling it. We also defined the oven and the fridge *critical* device class because they are prone to disaster if controlled maliciously.

[1]First implementation code at `https://github.com/giovannirosa/zash`
[2]Second implementation code at `https://github.com/giovannirosa/zash-cpp`
[3]Third implementation code at `https://github.com/giovannirosa/zash-ns-3`
[4]Dataset at `https://openshs.github.io/datasets/`

Table 5.1: Configured devices

Device	Device Class	Room	Type
Wardrobe	Non-critical	Bedroom	Active
TV	Non-critical	Living Room	Active
Oven	Critical	Kitchen	Active
Office Light	Non-critical	Office	Active
Office Door Lock	Critical	Office	Active
Office Door	Non-critical	Office	Active
Office Carpet	Non-critical	Office	Passive
Office Sensor	Non-critical	Office	Passive
Main Door Lock	Critical	House	Active
Main Door	Non-critical	House	Active
Living Light	Non-critical	Living Room	Active
Living Carpet	Non-critical	Living Room	Passive
Kitchen Light	Non-critical	Kitchen	Active
Kitchen Door Lock	Critical	Kitchen	Active
Kitchen Door	Non-critical	Kitchen	Active
Kitchen Carpet	Non-critical	Kitchen	Passive
Hallway Light	Non-critical	House	Active
Fridge Sensor	Critical	Kitchen	Active
Couch Sensor	Non-critical	Living Room	Passive
Bedroom Light	Non-critical	Bedroom	Active
Bedroom Door Lock	Critical	Bedroom	Active
Bedroom Door	Non-critical	Bedroom	Active
Bedroom Carpet	Non-critical	Bedroom	Passive
Bed Table Lamp	Non-critical	Bedroom	Active
Bed Sensor	Non-critical	Bedroom	Passive
Bathroom Light	Non-critical	Bathroom	Active
Bathroom Door Lock	Critical	Bathroom	Active
Bathroom Door	Non-critical	Bathroom	Active
Bathroom Carpet	Non-critical	Bathroom	Passive

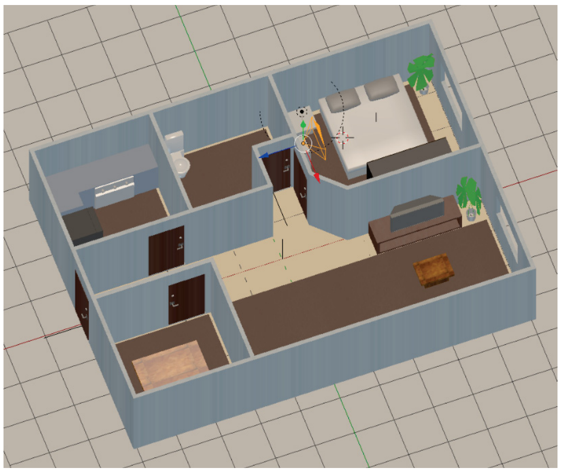


Figure 5.1: Home design (Alshammari et al., 2018)

5.2 TEST OF ALGORITHMS

This section comprises the correctness test of ZASH's algorithms implemented and executed in *Python*. Section 5.2.1 describes the implementation details. Section 5.2.2 presents the scenarios and metrics. Section 5.2.3 reports the results and analysis.

5.2.1 Implementation Details

The implementation[5] consisted of the modules from Section 4.2 with Object-oriented modeling, which has excellent modularity, flexibility, and reusability. We represented each module from the architecture in Object Oriented Programming (OOP). They interact with each other as described in Section 4.3. There is also an extra audit module to collect events from the simulation and calculate the metrics.

The code has the file structure presented in Figure 5.2. The zash file has the execution configuration and output details. The enums file contains enumerators practiced in the system, and the models file contains the models employed. Further, the modules directory includes the modules needed for the execution of ZASH, which are in the following structure: the audit directory, which comprehends the audit file that has the implementation to collect and calculate metrics in the simulation; the behavior directory, which comprises the configuration and notification files that implement these components; the collection directory, which involves the data and notification files that implement these components; and the decision directory, which embraces the authorization, activity, context, and ontology files that implement these components. The implementation of the Markov Chain is in the activity file.

[5]ZASH code at `https://github.com/giovannirosa/zash`

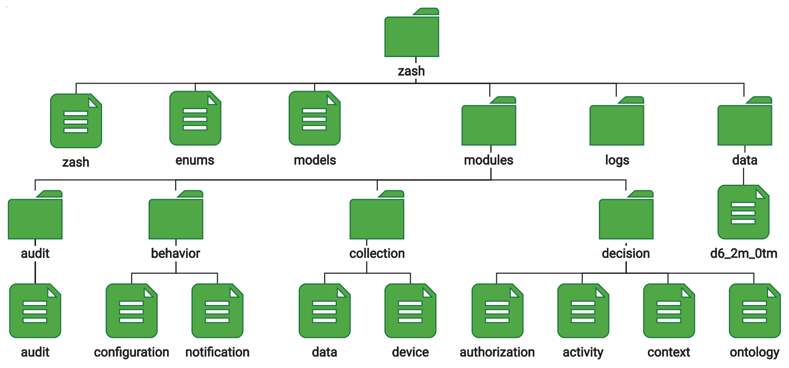


Figure 5.2: ZASH files in the Python implementation

The execution flows sequentially through the dataset records, simulating two months of Activities of Daily Living (ADLs). Table 5.2 displays how we configured users. We configured the ontologies as shown in Figure 4.6, the security layers as seen in Table 5.3, and the contextual factors as seen in Table 5.4. Ashibani et al. (2019) inspired these values, and we adjusted by experimentation to balance security and user experience with fewer proofs required.

Table 5.2: Configured users

	User 1	**User 2**	**User 3**	**User 4**	**User 5**
UL	Admin	Adult	Child	Child	Visitor
Age	Adult	Adult	Teen	Kid	Adult

Table 5.3: Configured security layers

	User Level				**Action**			**Device Class**	
Categories	Admin	Adult	Child	Visitor	Manage	Control	View	Critical	Non-critical
Security Level	70	50	30	0	40	20	0	30	0

Table 5.4: Configured context factors

	Time		**Localization**		**Age**			**Group**		**Access Way**		
Cat.	Common	Uncommon	Internal	External	Adult	Teen	Kid	Together	Alone	Requested	House	Personal
SL	20	10	30	10	30	20	10	10	0	30	20	10

5.2.2 Scenarios and Metrics

The scenario consisted of one user interacting with a smart home for two months, as it is in the selected dataset (Alshammari et al., 2018). We considered three threats in the evaluation of ZASH's correctness: **1)** valid users trying to access devices with no permission (violation of ontology rules); **2)** valid users trying to access devices in abnormal conditions (violation of context trust); and **3)** illegitimate users trying to access devices with stolen credentials (violation to activity pattern). The applied dataset influences the results as the activities might differ, leading to distinct probabilities in the Markov Chain.

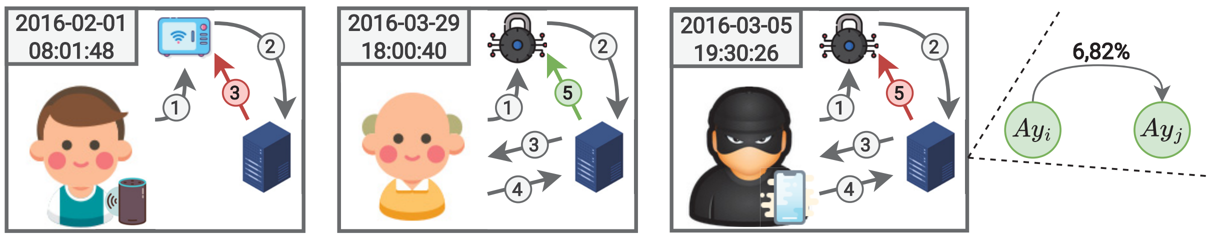
The metrics, shown in Table 5.5, assess the main behaviors proposed by ZASH. Ontology Fail (OF) measures the percentage of requests that failed in Ontology Manager and required proof of identity to verify authenticity. Similarly, Context Fail (CF) and Activity Fail (AF) evaluate the same as OF, but for the Context Manager and the Activity Manager, respectively. The Requests Granted (RG) measure the rate of granted requests, and the Requests Denied (RD) measure the rate of denied requests. We collected request fail events for ontology, context, and activity, block events, valid proofs, invalid proofs, total request number, granted request number, and denied request number to calculate the proposed metrics.

Table 5.5: Evaluation metrics

Description	Equation
Ontology Fail (OF) is the percentage of requests that failed in ontology evaluation (OF_{req}) out of total requests (T_{req}).	$OF = \frac{OF_{req}}{T_{req}} \times 100$
Context Fail (CF) is the percentage of requests that failed in context evaluation (CF_{req}) out of total requests (T_{req}).	$CF = \frac{CF_{req}}{T_{req}} \times 100$
Activity Fail (AF) is the percentage of requests that failed in activity evaluation (AF_{req}) out of total requests (T_{req}).	$AF = \frac{AF_{req}}{T_{req}} \times 100$
Requests Granted (RG) is the percentage of granted requests (G_{req}) out of total requests (T_{req}).	$RG = \frac{G_{req}}{T_{req}} \times 100$
Requests Denied (RD) is the percentage of denied requests (D_{req}) out of total requests (T_{req}).	$RD = \frac{D_{req}}{T_{req}} \times 100$

5.2.3 Results

We collected three use cases to validate the ZASH correctness, as exhibited in Figure 5.3. The *first* one, shown in Figure 5.3(a), is the request that happened on 2016-02-01 at 08:01:48 when a child user requested to control the oven (critical device) using a voice assistant (house access way). ZASH denied the action because a child cannot control a critical device in the configured ontologies shown in Figure 4.6. The *second* case, shown in Figure 5.3(b), occurred on 2016-03-28 at 08:04:10 when an admin user requested to manage the main door lock (critical device) by interacting directly with it. The expected trust is 100 because the admin UL_s (70) adds to the manage A_s (40), since the maximum expected trust is 100. However, the calculated trust from the instant context is 90, being requested access way (30), external localization (10), common time (20), alone (0), and adult (30). Therefore, ZASH asks for proof of identity from the user; since it is valid, the system grants the action. The *third* case, shown in Figure 5.3(c), is an impersonation attack that happened on 2016-03-05 at 19:30:26, where the attacker stole a personal device from an adult user and tried to control the main door lock (critical device) from an external location. The expected trust was 70 from adult UL_s (50) plus control A_s (20). The calculated trust was also 70 from personal access way (10), external localization (10), common time (20), alone (0), and adult (30). Although the Context Manager granted the action, the Activity Manager denied it because the P_{ij} equal to 6.82% was below the threshold of 10%. The results presented that every impersonation attack is prevented by ZASH, assuming the attacker cannot provide valid proof of identity belonging to the stolen user.



(a) Ontology case (b) Context case (c) Activity case

Figure 5.3: Use cases to verify ontologies rules, activities consistency, and context trust

Besides the individual use cases, we evaluated the general system behavior using the metrics. Table 5.6 displays the collected metrics from executions with variations in the request context and the user interacting with the smart devices. We executed simulations with a *baseline* for all requests with the context as personal access way, internal localization, and alone; from the admin user level, being an adult; and the action always as control. It provided an OF of 0% as all requests passed in Ontology Manager as expected since the admin user should have the highest privileges. It produced a CF of 1.77% due to requests performed at uncommon times. All non-blocked variations supplied 0.28% for AF since it depends only on activities. Considering the dataset was the same, the probabilities generated in the Markov Chain were also the same, thus producing the same percentage of requests that failed in the Activity Manager.

Table 5.6: Collected metrics from variations in context and user

Variations	OF	CF	AF	Blocked	RG	RD	Required Proofs
-	0%	**1.77%**	**0.28%**	No	100%	0%	**37**
External	0%	**100%**	**0.28%**	No	100%	0%	**216**
Adult	0%	0%	**0.28%**	No	100%	0%	**6**
Child/Visitor	**0.16%**	0%	0%	2016-02-01 08:05:52	**0.28%**	99.72%	0

Still, in the baseline, it had a RG of 100% as all 37 required proofs were valid. We executed the baseline context with *external localization*, providing similar results except for the CF, which was 100% as the context trust decayed and required 216 proofs for authenticity verification. Then, the baseline varied to *adult UL*, giving CF of 0%, once adult UL requires less trust than admin UL and requests only six proofs for the Activity Manager fails. Finally, the baseline ran for *child UL and visitor UL*, which resulted in the same metrics for both ULs, because the system blocked them by failing four times within 24 hours in Ontology Manager at 2016-02-01 08:05:52. ZASH just authorized 0.28% of requests before blocking the user and denying all the other 99.72%. The system assumes an attacker could not provide valid proof of identity, as detailed in Section 4.1. It is a complex task for the attacker to fake the proof since it is categorized as *Something You Are* (Lal et al., 2015). For now, ZASH is evaluating context and behavior, being a multi-factor authorization with *Somewhere You Are* from localization, *Something You Have* from access way, and *Something You Do* by evaluating activities. However, it could also mix these categories in the proof of identity phase to improve security by asking for a *Something You Know* challenge, for example.

5.3 SMART HOME NETWORK SIMULATION

This section evaluates ZASH implemented and executed in *ns-3*. Section 5.3.1 describes the implementation details. Section 5.3.2 presents the scenarios and metrics. Section 5.3.3 reports the results and analysis.

5.3.1 Implementation Details

We evaluated ZASH in a more realistic scenario considering network constraints not contemplated in the previous evaluation in Section 5.2. We implemented and analyzed ZASH in the *ns-3 network simulator (version 3.36.1)*[6], since it supports custom behavior with *C++* and *Python*, configurable topologies, and the adoption of well-known protocols, like Wi-Fi and 6LoWPAN. We run all simulations in the following hardware: *Intel® Core™ i7-8565U CPU @ 1.80GHz × 8, NVIDIA GeForce MX150/PCIe/SSE2, and 15,4 GiB RAM* with *Zorin OS 15.3, 64-bit*.

We also apply as input data for all scenarios the *SIMADL (Simulated Activities of Daily Living Dataset)*(Alshammari et al., 2018)[7] because it holds synthetic data collected on daily life activities with a significant amount of data in a well-structured form with 28 devices, as shown in Table 5.1. We selected the *d6_2m_0tm.csv* variant (167,211 lines of records after cleaning duplicates, 60 days, 1 line per second) for presenting the biggest amount of records compared

[6]ZASH code at `https://github.com/giovannirosa/zash-ns-3`
[7]Dataset at `https://openshs.github.io/datasets/`

to the other variants in the dataset. Device states are represented with 0 or 1; for example, a door with state 0 is closed and 1 is opened. There are eight passive and 21 active devices in the dataset. For instance, a door that needs direct user interaction to change its state means an active device; whereas a pressure sensor installed on the floor that detects users' movements without them explicitly interacting with the sensor means a passive device. Further, a state change on an active device indicates a user request, and a state change on a passive device denotes a user interaction with the environment. The home design comprehends one bedroom, one living room, one kitchen, one bathroom, one office, and one hallway. In Alshammari et al. (2018), the simulation uses this arrangement of a single-bedroom layout to create the dataset, representing a simplistic but realistic case. We defined all smart locks as a critical device class because an attacker could physically invade the house or even lock the user without consent by controlling it. The oven and the fridge were also configured as critical devices since they are prone to disaster when controlled maliciously by an attacker.

Each simulation round means an operation time of 57.48 days, corresponding to 4,966,270 seconds. We executed ten times per scenario to achieve more reliable results, and each round took around 14 minutes in chronological time. The network comprises 28 wireless sensors (STAs) distributed in the house as described in Table 5.7 and shown in Figure 5.4. One local server communicates with the sensors through a router (AP). The default data rate between the local server and the router is 100 Mbps with a delay of 1 ms. The Wi-Fi setting took ConstantRateWifiManager with data mode of HtMcs7, the control mode of ErpOfdmRate24Mbps, 802.11n standard, 2.4 GHz band, 20 MHz channel width, and channel 36. Besides, the AP considers a beacon interval of 4 s and a BSR lifetime of 0.8 s. The STAs operate with no active probing, a wait beacon time of 4.8 s, and an association request timeout of 20 s. The Short Guard Interval is enabled. We spread out the STAs in a house 40 m long and 28 m wide, thus 1,120 square meters. Their exact positions are described in Table 5.1 and remain the same throughout the simulation. The nodes work with IPv6, the local server IP is fe80::200:ff:fe00:1, the AP IP is fe80::200:ff:fe00:3, and STAs IPs range from fe80::200:ff:fe00:4 to fe80::200:ff:fe00:f.

Table 5.7: Nodes positions

Node	X (m)	Y (m)	Z (m)
Local Server	33	25	0
Router	39	23	0.5
Wardrobe	42	17	1
TV	43	22	0.6
Oven	12	19	0.5
Office Light	21	35	2.7
Office Door Lock	17.5	30	1.7
Office Door	17.5	31	1.5
Office Carpet	15	35	0
Office Sensor	17	37	1
Main Door Lock	8.5	31	1.7
Main Door	8.5	32	1.5
Living Light	37	30	2.7
Living Carpet	30	32	0
Kitchen Light	12	22	2.7
Kitchen Door Lock	18	24	1.7
Kitchen Door	18	25	1.5
Kitchen Carpet	15	21	0
Hallway Light	23	27	2.7
Fridge Sensor	8	23	1.8
Couch Sensor	44	27	0.5
Bedroom Light	31	14	2.7
Bedroom Door Lock	31	19	1.7
Bedroom Door	31	20	1.5
Bedroom Carpet	43.5	13	0
Bed Table Lamp	30.7	11	0.6
Bed Sensor	39	13	0.5
Bathroom Light	22	19	2.7
Bathroom Door Lock	29	16	1.7
Bathroom Door	29	17	1.5
Bathroom Carpet	20	16	0

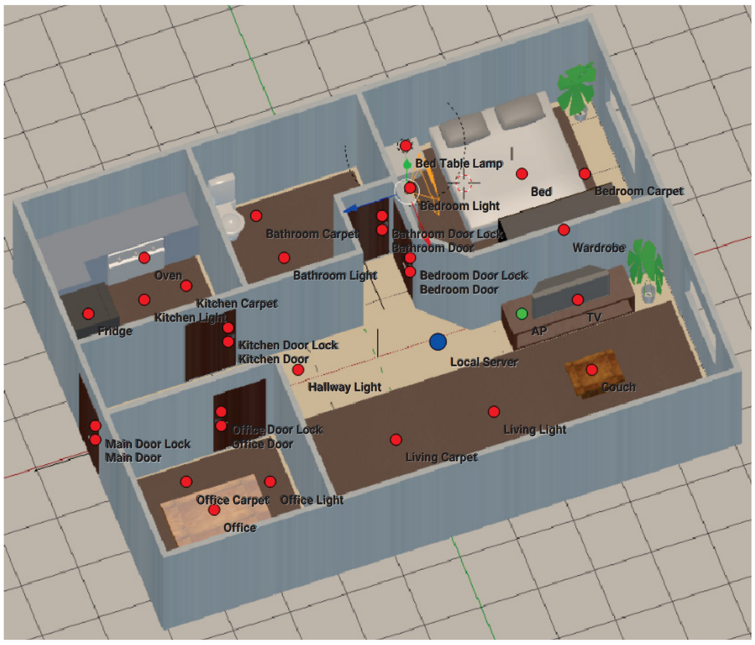


Figure 5.4: Devices deployment

The implementation has the file structure presented in Figure 5.5. The zash-simulator file has configurations for the network details, server structure, applications startup, messages schedule, and output. We inserted the ZASH module in the ns-3 source files. It follows the ns-3 standard with the doc, examples, helper, model, and test directories: the doc contains the documentation of the module; the examples include some usage examples of the module; the helper comprises the utils file with utility functions; the test consists of tests for the module; the model is composed of all the ZASH files. Further, the model directory includes the markov file, which has the Markov Chain implementation; the enums file, which contains enumerators practiced in the system; the models file, which contains the models of the system; the action directory, which comprehends the alteration and attack files that describe these actions; the audit directory, which involves the audit file that has the implementation to collect and calculate metrics in the simulation; the behavior directory, which contains the configuration and notification files that implement these components; the collection directory, which embraces the data and notification files that implement these components; and the decision directory, which incorporates the authorization, activity, context, and ontology files that implement these components.

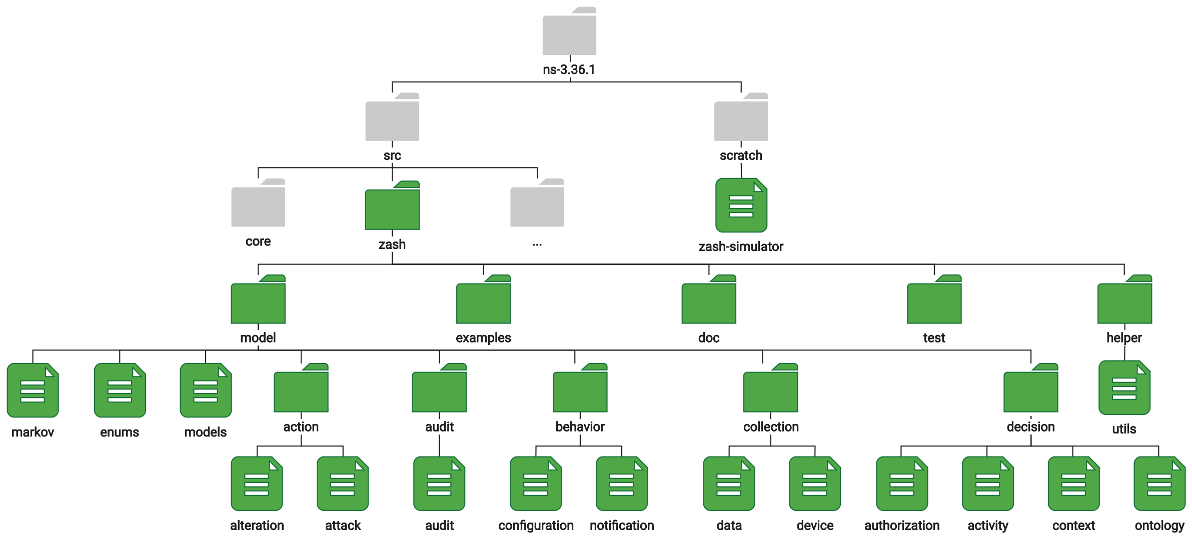


Figure 5.5: ZASH files inside the ns-3 structure

5.3.2 Scenarios and Metrics

We defined two security setting modes, hard (H) and soft (S), and their levels, as shown in Table 5.8 and Table 5.9, in order to compare the system results. These comparison modes reveal how the system reacts to different configurations like in Pan et al. (2019); Sikder et al. (2019a); Sikder et al. (2020). The hard mode holds a more strict configuration, and the soft mode holds less strict rules. Besides security levels, the hard mode has a block threshold of three attempts, a block interval of 24 hours, a build interval of 30 days, and a Markov threshold of 20%. In turn, the soft mode has a block threshold of six attempts, a block interval of 12 hours, a build interval of seven days, and a Markov threshold of 10%. We obtained these values heuristically by performing several tests. Combining all possible values for H, the highest security level expected achieved 100, the lowest expected 20, the highest calculated 100, and the lowest calculated 10, thus passing 38.70% of the possible combinations. In turn, for the soft mode, the highest security level expected reached 100, the lowest expected 0, the highest calculated 100, and the lowest calculated 30, thus passing 71.45% of the possible combinations.

Table 5.8: Configured security layers for the hard and soft modes

	User Level				Action			Device Class	
Categories	Admin	Adult	Child	Visitor	Manage	Control	View	Critical	Non-critical
Security Level (H)	70	50	30	10	40	20	10	30	10
Security Level (S)	70	50	30	0	40	20	0	30	0

Table 5.9: Configured context factors for the hard and soft modes

	Time		Localization		Age			Group		Access Way		
Cat.	Common	Uncommon	Internal	External	Adult	Teen	Kid	Together	Alone	Requested	House	Personal
SL (H)	20	0	20	0	20	10	0	10	0	30	20	10
SL (S)	20	10	30	10	30	20	10	10	0	30	20	10

We implemented the randomizations below in the simulation to reproduce more realistic scenarios. Moreover, there were three different normal actions in the simulation: the view action, simulated when the same state repeated sequentially from the dataset with a chance of 10% not to overload the simulation; the control action, simulated when there were changes in consecutive states from the dataset; the manage action, simulated by using randomizations of uniform distributions to select a valid time inside the dataset range and one device of the 28 possible. Further, if the picked device was available at that time, it was removed from the system, and the simulation ignored the next interactions with it. Otherwise, the simulation returned that device to the system, and its subsequent interactions are valid. The normal actions counted with randomization of uniform distribution to select an access way and a group.

Table 5.10: Discrete distributions for the attacks randomizations

	Localization		Access Way			Actions			User Profiles				
Cat.	Internal	External	Requested	House	Personal	Manage	Control	View	Admin	Adult	Teen	Kid	Visitor
Prob.	10%	90%	10%	10%	80%	10%	40%	50%	5%	10%	20%	25%	40%

Besides the normal actions, we implemented impersonation attacks to assess the system's security. In this attack, the attacker assumes the identity of another authentic user from the system and performs one of the actions in a specific context and device. We suppose the attacker obtained the user's credentials and tried to exploit it. This attack affects system confidentiality, authenticity, integrity, and non-repudiation and has consequences on users' security, safety, and privacy, causing physical, emotional, and material losses (Zheng et al., 2014; Geneiatakis et al., 2017; Mocrii et al., 2018; Al-Naji and Zagrouba, 2020). The attacks also employed randomizations of uniform distributions to select a valid time inside the dataset range and one device of 28 possible. Additionally, it applied discrete distributions, as shown in Table 5.10, for a more realistic behavior to determine a localization, an access way, an action, and the impersonated user. The only two restrictions to choosing a device for attacks were to be an *active* one if the selected action was *control* since it would not make sense to change the state of a passive device, and the selected device should *not be removed* at that time.

The metrics, shown in Table 5.11, measure the ZASH's robustness, efficiency, extensibility, and performance. They also assess the main characteristics proposed by ZASH to check if it satisfies the requirements of privacy perception, low latency in response, spatial and temporal locality, device extensibility, security against impersonation, device isolation, and security enforcement. The Privacy Risk (PR), Device Isolation (DI), and Access Control Enforcement (ACE) metrics evaluate the system's robustness. PR represents the risk to the user's privacy and is calculated based on two assumptions: more admin users in the system represent more ways an attacker can exploit to impersonate the most privileged User Level; more critical devices in the system mean more ways for an attacker to compromise the inhabitant's safety and security. DI relies on the assumption that more User Levels, Device Classes, and Actions in the system express more separation for accessing the devices. ACE supports that more interactions between users and smart devices without intermediaries mean more enforcement capacity of the system. The Access Control Response Time ($ACRT$), Access Control Distance (ACD), and Spatial and Temporal Locality (STL) metrics evaluate the system's efficiency. $ACRT$ is the time interval between the user request and the ZASH response and verifies whether the system response time is appropriate. ACD counts the number of hops from the smart devices to the server to check their proximity. STL considers the previous $ACRT$ and ACD metrics to know the time and space relation between the smart devices and the server. The Device Extensibility (DE) metric evaluates the system's extensibility and proves that ZASH can cope with the addition and removal of devices during its operation. The Attacks Denied Rate (ADR) metric evaluates the system's performance and measures system accuracy to prevent impersonation attacks.

Table 5.11: Evaluation metrics

Description	**Equation**
Privacy Risk (PR) is the number of admin users ($\|AU\|$) in the system multiplied by the number of devices with the critical class ($\|CD\|$) of the system.	$PR = \|AU\| \times \|CD\|$
Device Isolation (DI) is the number of user levels ($\|UL\|$) multiplied by the number of device classes ($\|DC\|$) multiplied by the number of actions ($\|A\|$).	$DI = \|UL\| \times \|DC\| \times \|A\|$
Access Control Enforcement (ACE) is the number of requests from the devices themselves, requested access way (RAW), divided by the number of requests from the home devices, home access way (HAW), added by the number of requests from the personal devices, personal access way (PAW). It can be simplified to the number of requests without intermediaries (NI) divided by the number of requests with intermediaries (I).	$ACE = \frac{RAW}{HAW + PAW} = \frac{NI}{I}$
Access Control Response Time ($ACRT$) is the arithmetic average of the difference between access request time (REQ) and access response time (REP) of all requests (n).	$ACRT = \frac{\sum_{i=1}^{n} REP_i - REQ_i}{n}$
Access Control Distance (ACD) is the arithmetic average of the distance between a smart device and the server, measured in hops (H), of all smart devices (n).	$ACD = \frac{\sum_{i=1}^{n} H_i}{n}$
Spatial and Temporal Locality (STL) is the Access Control Distance (ACD) multiplied by the Access Control Response Time ($ACRT$).	$STL = ACD \times ACRT$
Device Extensibility (DE) is the difference between the maximum (D_{max}) and minimum (D_{min}) number of devices over time.	$DE = D_{max} - D_{min}$
Attacks Denied Rate (ADR) is the percentage of avoided impersonation attacks (DI) out of the total number of impersonation attacks (TI).	$ADR = \frac{DI}{TI} \times 100$

5.3.3 Results

We simulated scenarios with the hard (H) and soft (S) modes, for different numbers of attacks, different numbers of device alterations by the manage action, good (G) and poor (P) network conditions, and different numbers of users. At first, we collected and analyzed the number of requests (R), the requests granted (RG), the requests denied (RD), the ontology fails (OF), the context fails (CF), the activity fails (AF), the number of required proofs (P), the valid proofs (VP), the invalid proofs (NP), and the number of blocks (B). Table 5.12 and all the following results display the mean values for 10 executions per scenario for more consistent and reliable results.

The values in Table 5.12 reveal all scenarios had similar requests (R), except those with different alterations, i.e., 10, 20, and 40. As expected, fewer requests happen when there are more alterations once more devices are removed. The scenario with 40 alterations had the smallest R value of 13,508.1. All scenarios exhibit a similar relation of requests granted (RG) and requests denied (RD), except for the scenario with H and 50 attacks, which had a higher RD value due to the higher number of blocks (B). That makes sense since this scenario has more attacks with a more strict configuration. Thus, it has a higher probability of generating blocks. The system promptly denied the blocked users' requests, contributing to the higher RD value. Further, all scenarios display similar ontology fails (OF), except for the scenario with only two users: one admin and one adult. Once these users have higher privileges than visitors and children, the ontology manager's decisions result in fewer denials.

Unlike those with S mode, most scenarios unveil similar context fails (CF). As expected, the less strict configuration for the trust calculation generated fewer fails in the context component. Besides, the scenarios with more alterations had a slightly higher CF, especially those with 40 alterations. The simulation can remove more devices during the execution, and the probability of requests with critical devices increases. Thus, the expected trust is higher, which generates more fails. Furthermore, the scenarios with S mode produce higher activity fails (AF) due to the lower CF. Since the context manager filters fewer cases than with H mode, the activity manager gets more denials. The number of required proofs (P) is proportional to the sum of fails in the three components, so the scenarios with S mode had fewer proofs required, and the scenario with more alterations had a slightly higher number. All scenarios demonstrate similar valid proofs (VP) and invalid proofs (NP) except those with S mode and 50 attacks. It had more invalid proofs because of the number of attacks, but it generated fewer blocks than H mode with 50 attacks due to the less strict configuration. With fewer blocks, more attacks continued to happen. Thus, the fails required more proof of identity, which were invalid.

Table 5.12: Profile of simulations from variations

Mode	# Att.	# Alt.	Net.	# Users	R	RG	RD	OF	CF	AF	P	VP	NP	B
S	**10**	**5**	**G**	**5**	18,521.2	99.97%	0.03%	0.01%	**0.36%**	**12.87%**	**214**	97.25%	2.75%	0
H	**10**	**5**	**G**	**5**	18,678.2	99.96%	0.04%	0.01%	55.32%	6.33%	340.5	97.89%	2.11%	0.4
S	**25**	**5**	**G**	**5**	18,742.1	99.92%	0.08%	0.03%	**0.35%**	**14.14%**	**222.4**	93.82%	6.18%	0.6
H	**25**	**5**	**G**	**5**	18,803	99.89%	0.11%	0.02%	55.31%	6.85%	348	95.81%	4.19%	2.5
S	**50**	**5**	**G**	**5**	18,805.6	99.82%	0.18%	0.05%	**0.35%**	**15.15%**	**233.7**	**89.91%**	**10.09%**	2.1
H	**50**	**5**	**G**	**5**	18,784.1	**95.73%**	**4.27%**	0.03%	54.57%	6.75%	337.8	94.72%	5.28%	**3.8**
H	**10**	**5**	**P**	**5**	18,745.1	99.96%	0.04%	0.02%	55.36%	7.05%	341.8	97.81%	2.19%	0.5
H	**10**	**10**	**G**	**5**	17,665.8	99.96%	0.04%	0.02%	56.44%	6.41%	343	97.96%	2.04%	0.4
H	**10**	**20**	**G**	**5**	15,117.9	99.95%	0.05%	0.02%	57.50%	6.03%	349.6	97.92%	2.08%	0.4
H	**10**	**40**	**G**	**5**	**13,508.1**	99.95%	0.05%	0.02%	**59.05%**	6.11%	**364.6**	98.05%	1.95%	0.4
H	**10**	**5**	**G**	**2**	18,963.3	99.95%	0.05%	**0.0005%**	54.93%	6.78%	339.2	98.41%	1.59%	1

The following sections present the results and analysis for four categories of metrics. Section 5.3.3.1 shows the evaluation of the system's robustness. Section 5.3.3.2 reports the efficiency of the system. Section 5.3.3.3 shows the assessment of the system's extensibility. Section 5.3.3.4 describes the system's performance.

5.3.3.1 Analysis of Robustness

The analysis of the robustness results comprehends the Privacy Risk (PR), Device Isolation (DI), and Access Control Enforcement (ACE) metrics. They reveal tendencies and help the administrator to configure the system for higher security. **Privacy Risk (PR)** helps the root user to know how likely it is to compromise the user's privacy given the number of critical devices (DC) and the number of admin users (AU). These properties do not depend entirely on the system management, meaning the root user should not reduce them artificially to have a smaller PR. On the contrary, the system manager should classify the devices and the users to represent the real world with high fidelity and check risks with PR.

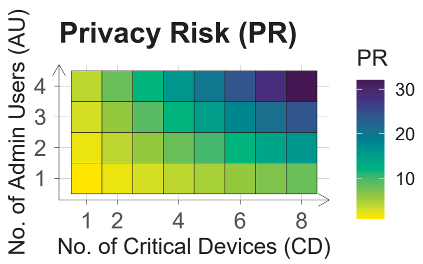


Figure 5.6: Privacy Risk for a certain number of critical devices and admin users

The more critical devices and admin users, the more risk to the privacy of all users in the system. We inferred it because the admin users have the most access privileges to the system and smart devices. Thus, if an attacker impersonates one of them, the privacy risk is higher than at a lower user level. In turn, the critical devices correspond to the most important ones that can cause a more significant impact once compromised than a lower device class. We made the simulation with one admin user and seven critical devices, resulting in a PR of 7. With one more admin user, the PR would double because there would be one more user with the highest privilege to exploit and cause device issues.

Device Isolation (DI) measures the number of security layers in the system. ZASH relies on user levels, device classes, and actions. These three layers help to protect and isolate the resources of the system, which are the smart devices. The higher number of layers, the higher DI; and the smart devices are more isolated and secure.

Table 5.13: Device Isolation for a certain number of user levels, device classes, and actions

UL	1	...	1	...	2	...	3	...	3	...	**4**
DC	1	...	3	...	2	...	2	...	3	...	**2**
A	1	...	3	...	2	...	3	...	3	...	**3**
DI	1	...	9	...	8	...	18	...	27	...	**24**

The ZASH simulations consisted of 4 user levels, two device classes, and three actions. That configuration is highlighted in Table 5.13 and produced a DI of 24, meaning there are 24 combinations for a resource to be isolated. The DI calculation favors a broader configuration regarding user levels, device classes, and actions because attackers will find it more difficult to penetrate these different spaces.

Access Control Enforcement (ACE) evaluates the relation between the requests with and without intermediaries. The intermediaries can be a smartphone, smart assistants, or other devices. The requests without intermediaries are those when the user interacts directly with the smart device. The ACE calculation favors this type of interaction because an attacker is less likely to follow this approach. Usually, attackers take advantage of stolen passwords and devices to interact indirectly and remotely with smart devices. Besides, the more intermediaries, the more vulnerable the system is once the attack surface is more extensive.

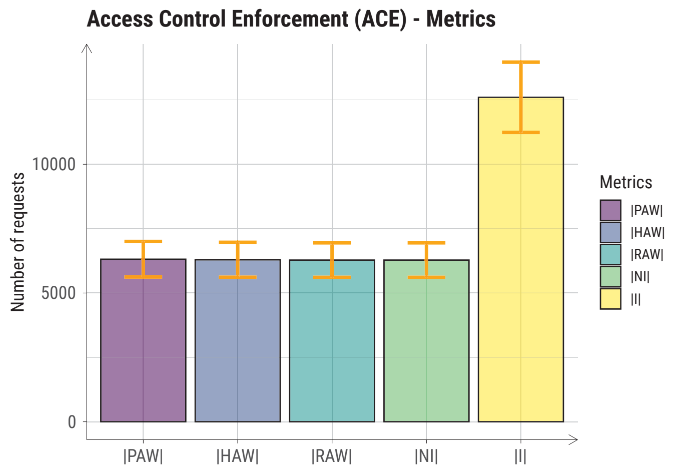


Figure 5.7: Access Control Enforcement metrics for all scenarios

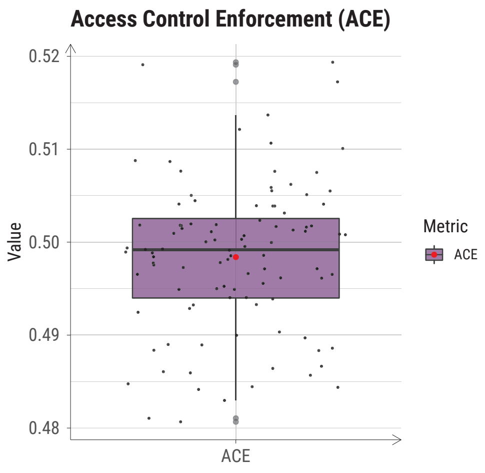


Figure 5.8: Access Control Enforcement for all scenarios

We counted with three access ways: *Personal*, *Home*, and *Requested*. The first two are indirect interactions since the Personal relates to access using a personal device, like a smartphone, and the Home relates to access using a device shared by all users in the house, like smart assistants or tablets. The Requested way relates to interactions directly with the smart devices. Since the simulations employed randomization of uniform distribution to select one access way for the request, the number of Personal Access Ways (PAW), number of Home Access Ways (HAW), and number of Requested Access Ways (RAW) were similar to all executions. Hence, the number of requests without intermediaries (NI), including only RAW, was the same proportion as the others. The number of requests with intermediaries (I), which includes the PAW and HAW, was double compared to NI. Thus, the ACE mean value was 0.4984, tending to 0.5, given the revealed conditions.

5.3.3.2 Analysis of Efficiency

The analysis of the efficiency results comprises the Access Control Distance (ACD), Access Control Response Time (ACRT), and Spatial and Temporal Locality (STL) metrics. They reveal the system's efficiency in performing access control. **Access Control Distance (ACD)** of ZASH had a fixed value of 2 since the messages departed from the smart devices through the Wi-Fi router and arrived in the local server. In comparison, the number of hops to reach a server in Google Cloud could be 11 (value obtained by executing traceroute for *google.com* from Porto Belo, SC, Brazil), increasing the exposition of the messages to external agents, as shown in Figure 5.9. This number can be even higher depending on whether the cloud servers are replicated in several regions and from where it is accessed. Further, the data packet hop limit or Time To Live (TTL) can be limited so it does not leak from the home network. The lower the ACD, the lower the probability of malicious entities intercepting data. Thus, this metric privileges a local network setup.

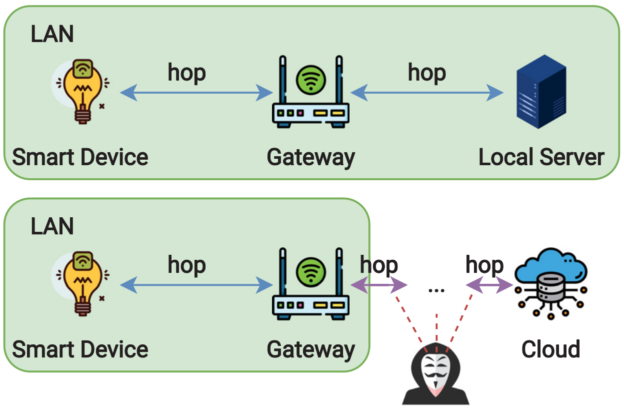


Figure 5.9: Low Access Control Distance vs. High Access Control Distance

We expanded the **Access Control Response Time (ACRT)** metric to analyze the results more deeply. The Access Control Response Time Blocked (ACRTB) is the response time for requests with a blocked user. The Access Control Response Time No Proof (ACRTNP) is the response time for requests that did not demand proof of identity. The Access Control Response Time Proof (ACRTP) is the response time for requests that require proof of identity.

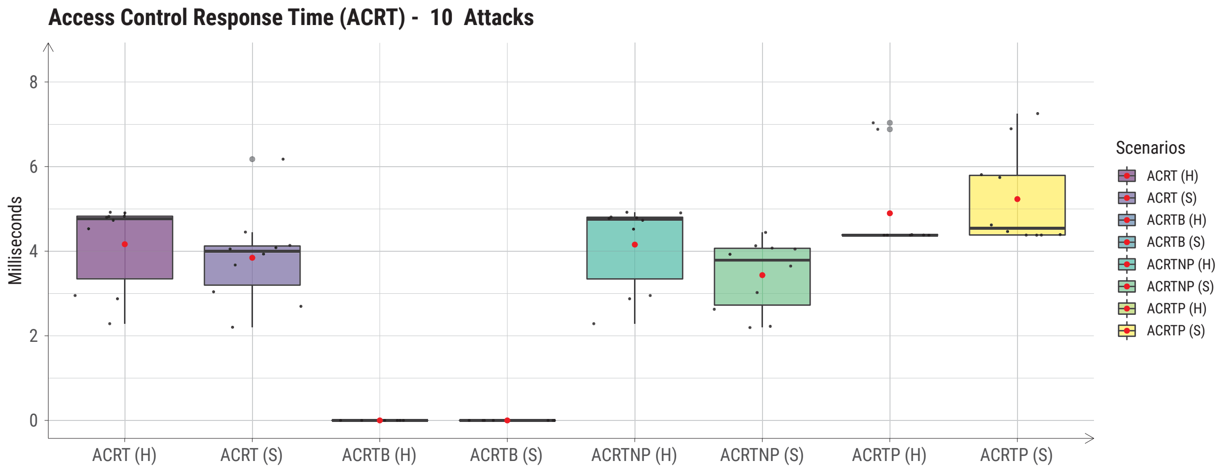


Figure 5.10: Access Control Response Time for ten attacks

The *first scenario* dealt with ten attacks during the simulations. Figure 5.10 shows ACRT(H) and ACRT(S) reached 4.16 ms and 3.84 ms. Both ACRTB were 0 ms because they did not present blocks for most executions. ACRTNP(H) and ACRTNP(S) achieved 4.16 ms and 3.43 ms; and ACRTP(H) and ACRTP(S) attained 4.90 ms and 5.23 ms. Those results unveil no significant difference for all metrics between H and S. The *second scenario* reckoned 25 attacks during the simulations. As shown in Figure 5.11, ACRT(H) and ACRT(S) achieved 3.73 ms and 3.87 ms. ACRTB(H) and ACRTB(S) were 2.12 ms and 0.38 ms. ACRTNP(H) and ACRTNP(S) obtained 3.29 ms and 3.42 ms; and ACRTP(H) and ACRTP(S) resulted in 4.98 ms and 4.93 ms. ACRTB(H) presented a higher value due to more interactions with blocked users, and others unveil no significant difference between H and S. The *third scenario* faced 50 attacks during the simulations. As shown in Figure 5.12, ACRT(H) and ACRT(S) were 4.04 ms and 4.03

ms. ACRTB(H) and ACRTB(S) reached 2.28 ms and 2.12 ms. ACRTNP(H) and ACRTNP(S) attained 3.58 ms and 3.74 ms; and ACRTP(H) and ACRTP(S) achieved 4.63 ms and 5.00 ms. Those results unveil no significant contrast for all metrics between H and S.

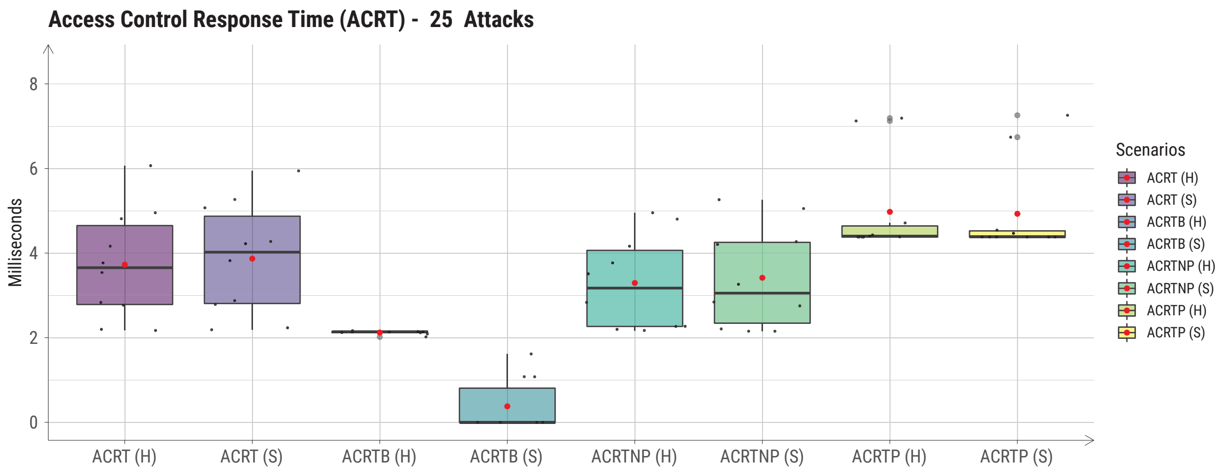


Figure 5.11: Access Control Response Time for 25 attacks

The results also revealed the number of attacks does not significantly affect the ACRT. They play an essential role in ACRTB since more attacks mean a higher chance of blocking users, thus more blocked responses. The time does not differ because it was just a matter of existing more blocks and not interfering with the response time. Besides, we see a tendency for requests without the proof required to be faster, as expected, since they need two fewer messages exchanged between devices and the server.

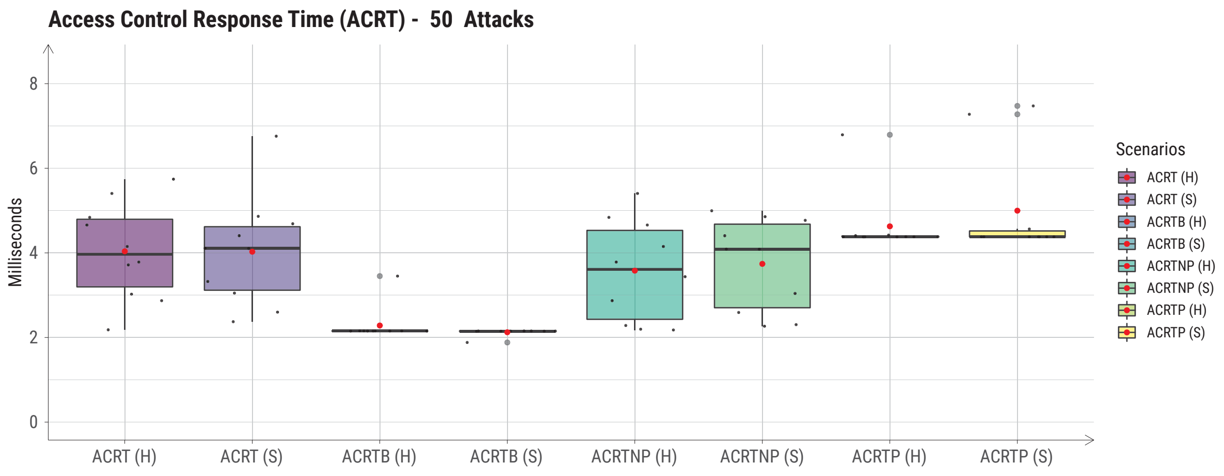


Figure 5.12: Access Control Response Time for 50 attacks

The *fourth scenario* compared results for good (100Mbps and 1ms) and poor (10Mbps and 50ms) network conditions with ten attacks. As shown in Figure 5.13, ACRT(G) and ACRT(P) acquired 4.16 ms and 153.77 ms. ACRTB(G) and ACRTB(P) were 0 ms and 12.54 ms. ACRTNP(G) and ACRTNP(P) obtained 4.16 ms and 153.45 ms; and ACRTP(G) and ACRTP(P) achieved 4.90 ms and 213.98 ms. As expected, the cases with good and bad network conditions presented the most significant contrast once the throughput and latency directly impacted the device's communication time with the server. Nevertheless, for all cases, the value was within the ideal reference for an SHS, even with bad network conditions. As exposed in Section 3.5, according to Google (2021), acceptable latency for a user interacting with Google Assistant is ideal when less than 200 ms, OK between 2 s and 5 s, and not acceptable if higher than 5 s.

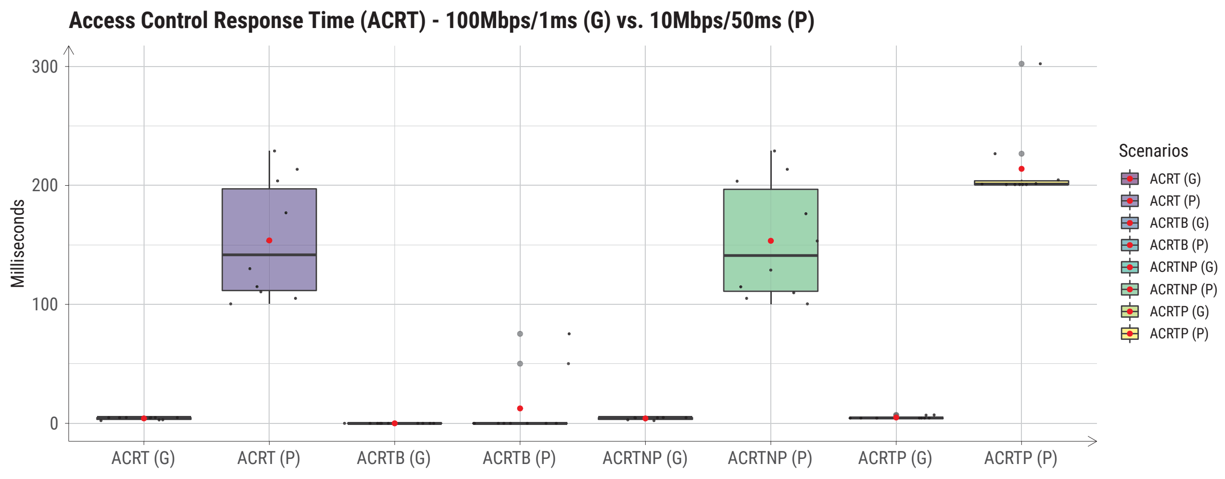


Figure 5.13: Access Control Response Time for good and poor network conditions

The *fifth scenario* analyzed the outcomes for 5, 10, 20, and 40 alterations. As shown in Figure 5.14, ACRT(5), ACRT(10), ACRT(20), and ACRT(40) reached 3.31 ms, 3.28 ms, 3 ms, and 3.12 ms. Although the number of alterations did not significantly impact the ACRT, there was a tendency to decrease it slightly with fewer devices due to the reduced requests.

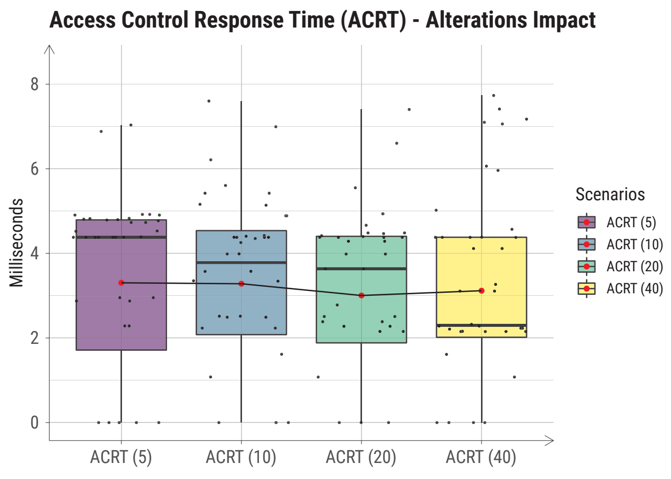


Figure 5.14: Access Control Response Time for different numbers of alterations

We calculated **Spatial and Temporal Locality (STL)** based on the previous metrics results from the H scenarios since it depends on the ACD and ACRT values. *We decided to use the median value of the ACRT boxplot for calculations instead of the general mean because it* *discards the outliers. Since ACD was the constant value of two for all scenarios, we omit its value for the next explanations.* As shown in Figure 5.15, for the scenarios with a different number of attacks: STL(10) acquired 8.32 as ACRT(10) was 4.16 ms; STL(25) attained 7.46 since ACRT(25) was 3.73 ms; and STL(50) resulted in 8.08 once ACRT(50) was 4.04 ms. Although there was no significant difference, STL tends to decrease with more attacks since it should present more blocks and reduce the ACRT value because requests with blocked users are faster.

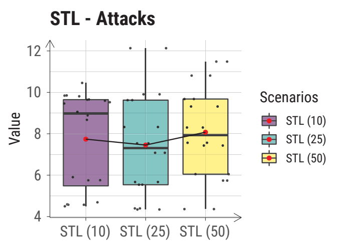


Figure 5.15: Spatial and Temporal Locality for attacks

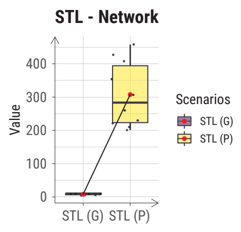


Figure 5.16: Spatial and Temporal Locality for network conditions

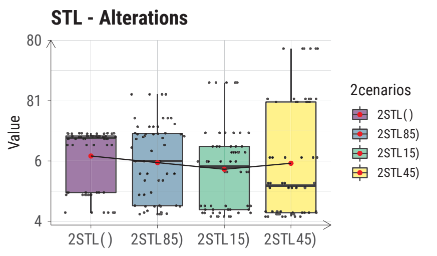


Figure 5.17: Spatial and Temporal Locality for alterations

Figure 5.16 shows the scenarios with different network conditions. STL(G) achieved 8.32 as ACRT(G) was 4.16 ms, and STL(P) reached 307.53 once ACRT(P) was 153.77 ms. Similarly, from the ACRT analysis, the network conditions play a significant role in the STL value due to the impact of ACRT. Figure 5.17 shows the results for the scenarios with different numbers of alterations. STL(5) acquired 8.32 since ACRT(5) was 4.16 ms. STL(10) resulted in 7.96 once ACRT(10) was 3.98 ms. STL(20) attained 7.45 given that ACRT(20) was 2.48 ms; and STL(40) reached 7.85 as ACRT(40) was 3.93 ms. We applied the same analysis from ACRT to STL since the number of alterations does not influence STL significantly. Still, there was a tendency to decrease it slightly and have a lower amplitude with fewer devices due to the reduced number of requests.

5.3.3.3 Analysis of Extensibility

The analysis of the extensibility results contains the Device Extensibility (DE) metric and aims to reveal how extensible the system performs access control. We measured **Device Extensibility (DE)** with the simulation of the addition and removal of devices. The system should continue working normally after a new device is added or an existing one is removed. The main impact of that alteration remains in the Markov Chain since the current state length is changed. We can think of two strategies to cope with that situation. The first is to ignore this change by filling older states to reach the necessary length in case of addition and leaving old states untouched in case of removal. The second one is to reset the Markov Chain and make it start building again when the current state length changes. The first option allows the Activity Manager to keep working but sacrifices the accuracy once it fills states with an artificial or old value. The second option consists of a more drastic approach, forcing the Activity Manager to return to the building state. Although it would consider only real interactions, the security could be compromised as the system would skip the activities verification in this period. ZASH employed the first strategy to favor security and not reset the Markov Chain.

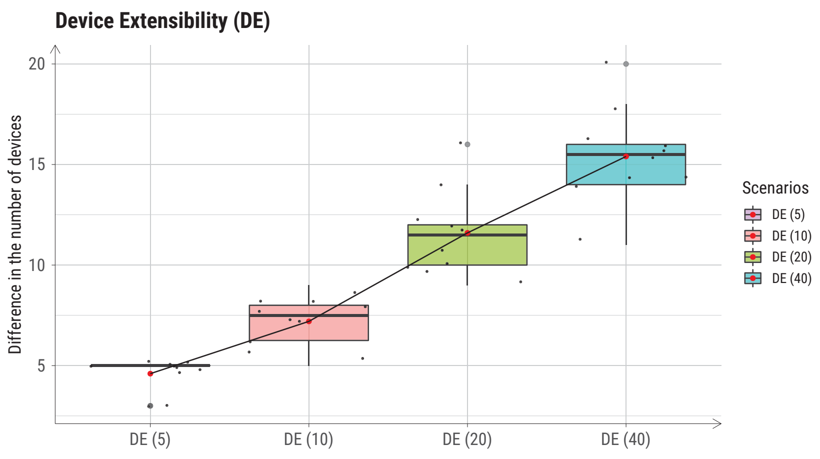


Figure 5.18: Device Extensibility for different numbers of alterations

We simulated scenarios with 5, 10, 20, and 40 alterations. As shown in the Figure 5.18, DE(5), DE(10), DE(20), and DE(40) resulted in 4.6, 7.2, 11.6, and 15.4. We expected this linear progression since the more alterations, the higher the probability of removing more devices.

5.3.3.4 Analysis of Performance

The analysis of the performance results involves the Attacks Denied Rate (ADR) metric and aims to reveal how performant the system's access control is. We expanded the **Attacks Denied Rate (ADR)** metric to analyze the results more deeply. The Denied Impersonations Building (DIB) is the percentage of denied impersonations while ZASH was building the Markov Chain. The Denied Impersonations Blocked (DIX) is the percentage of denied impersonations while the impersonated user was already blocked. The Successful Impersonations Building (SIB) is the percentage of successful impersonations while ZASH was building the Markov Chain. The Successful Impersonations Proof (SIP) is the percentage of successful impersonations while the proof of identity is still valid for the impersonated user. This last case is rare and will only happen when the attacker impersonates a user and interacts with the system within the interval of a valid proof of identity, which was 10 minutes for our simulations.

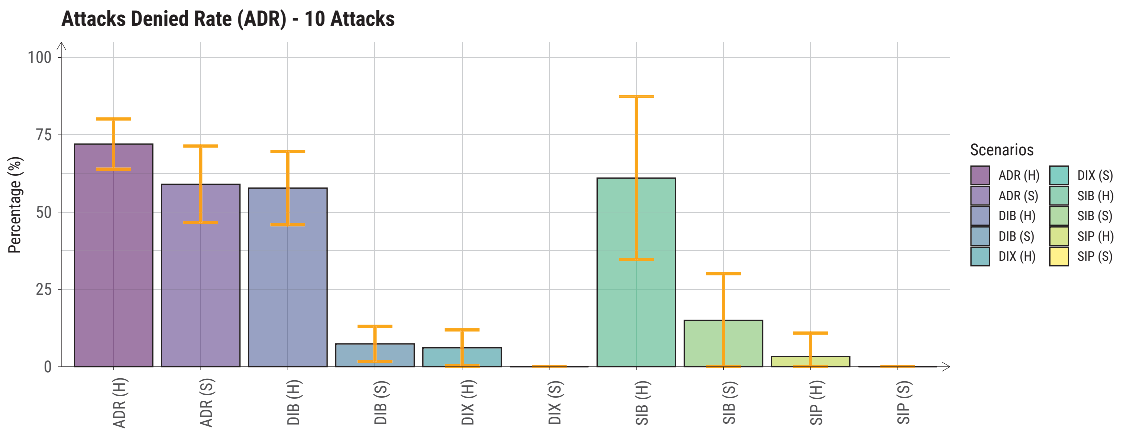


Figure 5.19: Attacks Denied Rate for ten attacks

The *first scenario* reckoned ten attacks during H and S simulations. As shown in Figure 5.19, ADR(H) and ADR(S) were 72% and 59%. DIB(H) and DIB(S) reached 57.77% and 7.36%. DIX(H) and DIX(S) obtained 6.11% and 0%. SIB(H) and SIB(S) achieved 61% and 15%; and SIP(H) and SIP(S) resulted in 3.33% and 0%. The *second scenario* imposed 25 attacks during H and S simulations. As shown in Figure 5.20, ADR(H) and ADR(S) obtained 86% and 56.8%. DIB(H) and DIB(S) acquired 46.56% and 5.50%. DIX(H) and DIX(S) achieved 43.53% and 6.93%. SIB(H) and SIB(S) attained 65.67% and 25.69%; and SIP(H) and SIP(S) reached 2% and 0.77%. The *third scenario* dealt with 50 attacks during the simulations for H and S. As shown in Figure 5.21, ADR(H) and ADR(S) resulted in 85.80% and 69.2%. DIB(H) and DIB(S) were 44.94% and 6.64%. DIX(H) and DIX(S) reached 68.23% and 36.94%. SIB(H) and SIB(S) achieved 73.40% and 23.93%; and SIP(H) and SIP(S) obtained 3.41% and 0%.

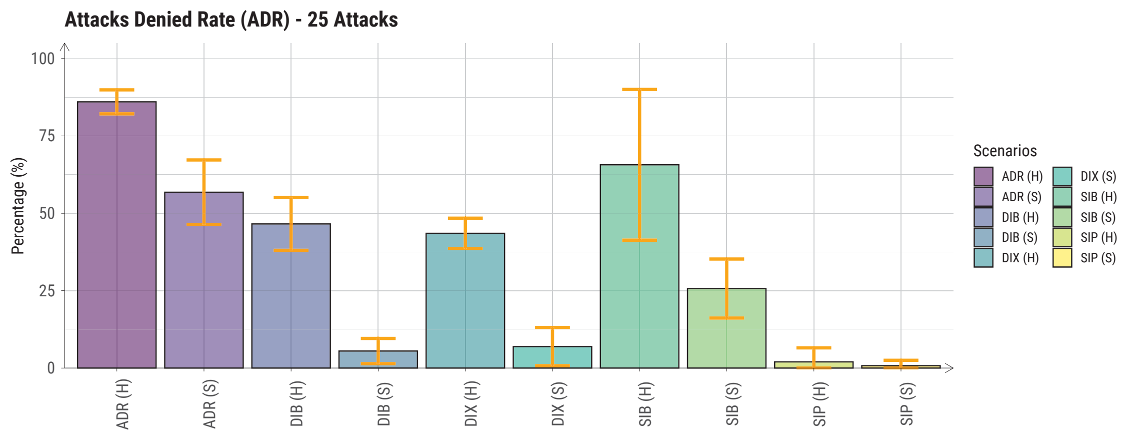


Figure 5.20: Attacks Denied Rate for 25 attacks

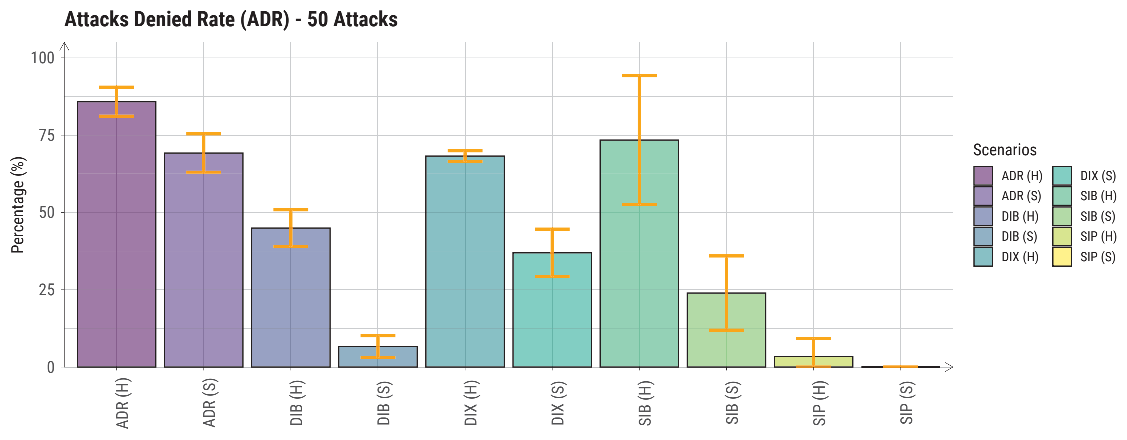


Figure 5.21: Attacks Denied Rate for 50 attacks

The results unveil that H presents higher rates for all metrics because it has a more strict configuration than S. Besides, DIB(H) is higher due to the extended build period and more strict rules. Furthermore, SIB(H) is higher due to the extended build period. Moreover, SIP(H) is slightly higher because strict rules generate more proofs asked, which are valid for 10 minutes. As expected, there was a remarkable influence in the DIX metric because the more attacks, the higher the number of blocks; thus, the system promptly denied more impersonations for blocked users, affecting ADR.

The *fourth scenario* tackled ten attacks on the simulations for H with 5, 10, 20, and 40 alterations. As shown in Figure 5.22, ADR(5), ADR(10) reached 72%, 73%, and both ADR(20) and ADR(40) were 74%. DIB(5), DIB(10), DIB(20), DIB(40) acquired 57.77%, 41.86%, 49.23%, and 53.10%. DIX(5), DIX(10), DIX(20), DIX(40) were 6.11%, 8.25%, 6.11%, and 8.19%. SIB(5), SIB(10), SIB(20), SIB(40) came to 61%, 45.33%, 29%, and 63.33%; and SIP(5) attained 3.33%, and SIP(10), SIP(20), and SIP(40) were 0%. We cannot make correlations between the metrics and scenarios. However, ZASH keeps good security even with more alterations considering mainly the ADR values.

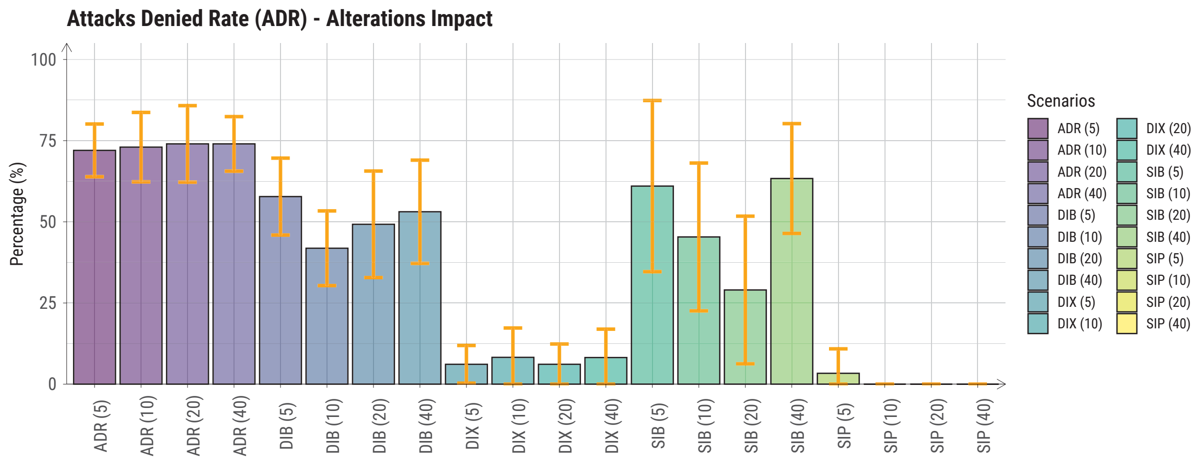


Figure 5.22: Attacks Denied Rate for different numbers of alterations

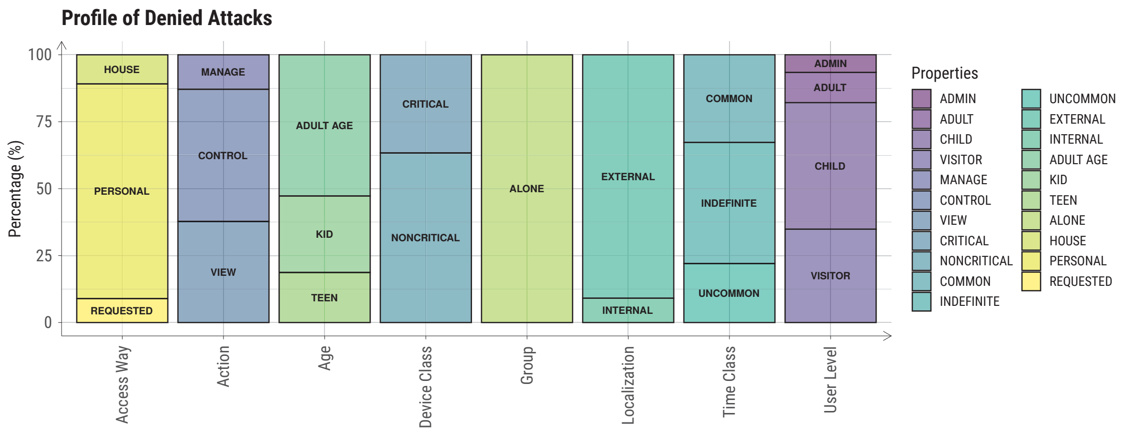


Figure 5.23: Profile of denied attacks for all scenarios

The denied attack profile for all scenarios is displayed in Figure 5.23. The Access Way was 8.94% Requested, 80.18% Personal, and 10.88% House. The Action reached 37.79% View, 12.87% Manage, and 49.34% Control. The Age attained 18.73% Teen, 28.53% Kid, and 52.74% Adult. The Device Class acquired 63.35% Noncritical and 36.65% Critical. The Group obtained 100% Alone. The Localization resulted in 9.09% Internal and 90.91% External. The Time Class achieved 22.05% Uncommon, 45.22% Indefinite, and 32.73% Common. The User Level reached 34.86% Visitor, 47.26% Child, 11.28% Adult, and 6.60% Admin. As expected, the results are similar to the discrete distributions for the attacks randomizations exposed in Table 5.10 because most impersonations were denied and followed these distributions.

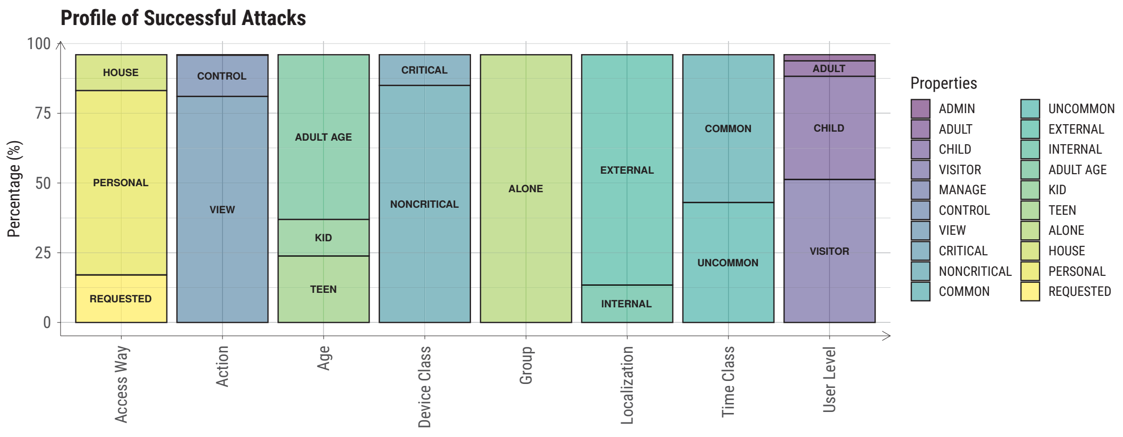


Figure 5.24: Profile of successful attacks for all scenarios

The *profile of successful attacks* for all scenarios includes the characteristics of its requests. The Access Way was 37.07% Requested, 66.07% Personal, and 12.86% House. The Action resulted in 81.04% View, 0.16% Manage, and 14.80% Control as the View action was less strict than the others. The Age reached 23.85% Teen, 13.10% Kid, and 59.05% Adult. The Device Class acquired 84.98% Noncritical and 11.02% Critical. The Group obtained 100% Alone. The Localization came to 13.45% Internal and 82.55% External. The Time Class achieved 43.03% Uncommon and 52.97% Common; and the User Level attained 51.32% Visitor, 36.95% Child, 5.53% Adult, and 2.20% Admin. Thus, the Visitor and Child users counted for 88.27% of the successful attacks. It makes sense since these user levels expected smaller trust values. Ideally, the Visitor user should be temporary access to reduce the exposition to attackers. We can configure the Child user to expect a higher trust value. However, it will ask for more proof of identity, which can be cumbersome, especially for children.

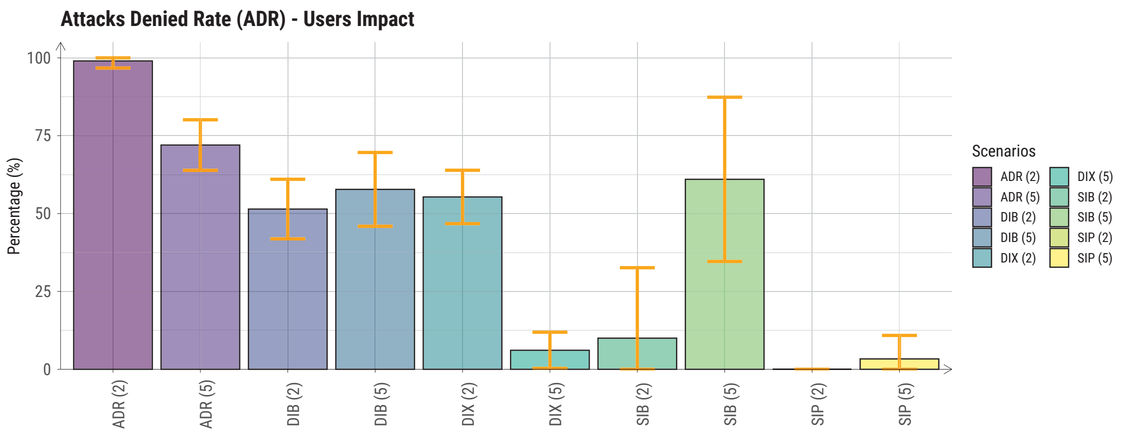


Figure 5.25: Attacks Denied Rate for different numbers of users

Due to previous results, we executed a scenario with *only admin and adult users* to check the attacks' impact. As shown in Figure 5.25, comparing scenarios with two and five users: ADR(5) and ADR(2) were 72% and 99%. ADR got much higher by removing the visitor and child users because the remaining admin and adult users require a higher trust value. DIB(5) and DIB(2) achieved 57.77% and 49.56%. DIX(5) and DIX(2) reached 6.11% and 52.56% due to more blocks generated by more strict users. SIB(5) and SIB(2) attained 61% and 10% due to more attacks prevented by the higher expected trust from the context; and SIP(5) and SIP(2) obtained 3.33% and 10%.

5.4 SUMMARY

This chapter described a complete evaluation of the proposed ZASH system. At first, we employed the necessary tools to implement and simulate a daily activities sequence to validate ZASH, mainly the decision-making. The dataset is publicly available and generated using a novel approach to synthesize the activity sequence from real user interaction in a virtual environment containing several sensors and smart devices. These dataset characteristics helped to simulate a realistic scenario to verify ZASH behavior by extracting individual cases and collecting general metrics. The preliminary results showed a promissory perspective that needed further investigation in a realistic home network.

After that, we implemented ZASH in more realistic scenarios using the ns-3 network simulator to simulate a Smart Home System with three normal actions (view, control, manage) and attacks. We applied randomizations of uniform and discrete distributions to simulate realistic situations for attacks and alterations. We checked that ZASH respects users' privacy, has a fast response time, copes with adding and removing devices, blocks most attacks, isolates smart devices, and enforces access control for all types of interactions. Besides, it is customizable and suitable for different configurations.

6 CONCLUSION

More people have been gaining access to smart gadgets, like smart bands, smart lights, smart locks, etc. However, privacy and security are still significant concerns for users, considering the exploitation of security breaches leads to physiological and financial harm for the affected inhabitants. Therefore, bringing a diverse range of these gadgets inside people's houses requires a robust layer of security to support the heterogeneity of technologies, user-device interactions, and services. The state-of-the-art security systems for smart homes have revealed some limitations on current works. Namely, the requirement for historical data to start the system operation, the dependence on external networks and resources, and the dependence on intermediaries to enforce access control. These points led to the definition of requirements for smart home security systems: privacy perception, low latency in response, spatial and temporal locality, device extensibility, security against impersonation, device isolation, and security enforcement.

This work has proposed the ZASH system to secure an SHS against impersonation attacks supported by continuous authentication, context-aware, and behavior-based access control with Zero Trust. ZASH endures the smart home heterogeneity of technologies and does not rely on external service or cloud computing to provide home security. It works in a local server that receives all device requests to ensure the ontology rules, the context trustworthiness, and the activity consistency. When the context trust is below expected or activities sequence probability is below a threshold, ZASH requires proof of identity of the *Something You Are* category to assure user authenticity. That makes the system learn new behavior and also blocks impersonation attacks. The system relies on multiple security layers with user levels, actions, and device classes to mitigate the control area of any undetected impersonation attack.

We implemented and evaluated ZASH first in a preliminary version to validate its capacity to prevent and mitigate impersonation attacks using instant context information and continuously adapting to users' behavior. Then, we also implemented it in a realistic home network with some assertions to measure the impact on user privacy, system reliability, access control response time, device extensibility, resistance to attacks, device isolation, and access control enforcement. According to results, ZASH protects users' privacy, responds quickly, copes with adding and removing devices, blocks most attacks, isolates smart devices, and enforces access control for all interactions. ZASH stops 72% with a scenario with children and visitors users and 99% of the impersonation attacks with only admin and adult users and a proper configuration for the SHS characteristics since each home has different users and devices. The access control response time is around 4.16 ms for good network conditions and up to 153.77 ms for bad network conditions, proving the ZASH decision time is within the ideal according to the Google Assistant parameters.

The main problem raised at the start of the research about how to prevent and mitigate unauthorized access on the smart devices in SHS caused by stolen credentials to protect the data and physical privacy of the inhabitants is answered by applying continuous authentication, context-aware and behavior-based access control with the Zero Trust paradigm. The first question about the common approaches for access control in smart homes and which of them could be the most effective to prevent and mitigate impersonation attacks is answered by authorization, Intrusion Detection System (IDS), and continuous authentication. The last one is the most appropriate to counter impersonation attacks once it continually verifies the users' authenticity. The second question about how to protect all communication interactions and access ways between users and smart devices, including digital and physical means, is answered by enforcing access control on the final smart device and not relying on intermediaries to prevent the user from bypassing the security system.

The third question raised about among the approaches listed in the first question, what are the techniques that can be employed to ensure operation from day zero is answered by a model that must work from day zero with instant data and then, as time passes, is improved with another module that counts on historical data. The fourth question about how to make the access control for smart homes self-sufficient regarding data and processing independent of external resources is answered by using edge computing with all sensors, data, and processing owned by inhabitants, with minimum collected data. The fifth question about how robust and effective access control can be for smart homes with zero trust continuous authentication against impersonation attacks is answered by all the results from Chapter 5. Thus, we verified ZASH is viable, prevents impersonation attacks, and fulfills the requirements for a security system for smart homes.

6.1 FUTURE WORKS

Despite answering the questions and addressing the main problem raised at the start of the research, there are still some paths to explore that were out of the scope of this research. The additional questions lead to future works and comprise approaches that could improve the impersonations blockage rate, test other system architectures, check for simultaneous interactions, assess particular users' treatment, analyze with qualitative metrics, and interact with other societal entities to reach more comprehensive security for the users.

1. *Improvement of impersonations blockage rate according to context*
   The blockage of impersonation attacks is essential to protect users' privacy, security, and safety. However, it is hard to represent complex scenarios from the real world as smart homes. Markov Chain can be turned into Hidden Markov Chain using device states as transitions and user activities as emissions to make the model more precise, as proposed by Fan et al. (2017). A new module to identify and describe user activity from device states would be needed. The activities threshold can be investigated to adapt depending on the request, similar to the context expected trust value, thus being more aware of the context but with higher complexity. The access control could include an auxiliary module to adjust its parameters automatically using an optimization algorithm on the fly to minimize false positives and maximize the impersonations block. The ideal or minimum number of smart devices must be investigated to keep a fair impersonation attack blockage rate, as it is still uncommon for people to have many sensors at home. A machine learning approach could calculate the best combination of ontologies and context weights configuration. Strategies to minimize the impact of adding and removing devices in the Markov Chain is another research possibility.

2. *System architecture organization for robustness and scalability*
   Smart home access control systems usually work with a centralized architecture to benefit from having a central point to gather info and perform the reasoning process promptly. However, the centralized architecture has disadvantages as a central node failure causes the entire system to fail; it cannot scale horizontally and can present bottlenecks as it depends on only one central node. Schemes to overcome the vulnerabilities of a centralized server considering the constraints of a local network and smart devices with serious energy, processing power, and storage restrictions, can be examined by testing decentralized or distributed architectures, as proposed by Gajewski et al. (2019). As

explained by Baran (1964), the decentralized architecture consists of multiple central nodes, leading to a more tolerant of faults system, better performance, and a more diverse and flexible system. However, decentralized systems are still prone to the same security and privacy risks to users as centralized systems, have higher maintenance costs, and can present inconsistent performance when not correctly optimized. It could be implemented in smart homes to spread the storage and processing by having one server by room, for instance. However, that would cause a problem if we need to gather the data for all rooms and process it. The distributed architecture does not have central nodes, as each entity in the system shares the same responsibility. It is fault-tolerant, transparent, secure, scalable, and promotes resource sharing. However, it is more challenging to deploy and presents higher maintenance costs. The constraints of smart devices make it challenging to implement a distributed architecture, but they can have more capacity as technology advances.

3. *Simultaneous interactions between users and smart devices*
   The investigated model considers only one user interacting with the smart home. Nevertheless, a smart home usually has multiple users simultaneously interacting with the smart devices. Thus, the system could be tested in a multi-user scenario to evaluate the impersonations blockage rate and user experience to verify its viability and acceptance. The complexity of these cases can lead to conflicts when more than one user interacts with the same device, or in the case of impersonation, the same user interacts with the same or different devices. These conflicts can be dealt with well-defined policies as proposed by Sikder et al. (2019b). Another challenge is collecting data from different people in the same environment. The system must recognize who is generating the data to track the behavior of that specific user. Vision-based approaches usually raise serious privacy concerns and require the presence of line-of-sight. Techniques to preserve users' privacy using radio waves to identify people can be tested as proposed by Yang et al. (2020).

4. *Special users treatment according to user type profiles*
   The results showed that adult and administrator users' impersonation attack blockage rate was high. However, it also showed that the system must treat visitors and child users differently since their blockage rate was below adults and administrators. As possible solutions, the system should regard temporary access for the visitor, and other techniques can be explored for child users, as they have specific requirements and are more vulnerable to the intrinsic characteristics of this type of user. Visitors usually will be rotating, so a profile based on their behavior will not have a proper time and sampling. The refined configuration of the ontologies will better constrain the child's profile, as their behavior when interacting with the smart devices also poses a safety risk. Sun et al. (2021) presented an extensive survey through 23 semi-structured interviews with parents who are smart home technology adopters to find six factors that shaped parents' safety perceptions and mitigation strategies, including parenting style, parents' tech-savviness, parents' trust in tech companies, children's age and developmental differences, news media, and device characteristics.

5. *Analysis with qualitative metrics related to users behavior*
   The evaluation only tested the model's correctness, references, efficiency, extensibility, and performance. However, qualitative metrics could also reveal exciting results. For instance, users' engagement can be investigated to check whether they will keep using it

over time or prefer to go over the access control for convenience. Further, a big challenge is configuring the proofs requirement and block settings to maximize the attack denial and user experience. Park et al. (2017) explores the key determinants of user acceptance of IoT technologies in a smart home environment and investigates a research model integrated with five potential user factors and a technology acceptance model. It shows that three positive motivations, compatibility, connectedness, and control, and a negative hindrance, cost, are significant determinants of the technology acceptance behavior of users. Chalhoub et al. (2020) designed and conducted six in-depth interviews with employees of a large smart home company in the United Kingdom (UK). It recommends that design teams be diversified to include all domains, educate and motivate design teams about UX in security and privacy, address the UX of hardware products, and develop innovative solutions to comply with GDPR.

6. *Integrated security to trigger defensive actions*
   Despite blocking the impersonation attacks, the system must integrate with external entities to respond appropriately to the attack. Automated responses to attacks and blocks can be studied, for example, integrating with police intelligence, notifying specific people, or even triggering actuators like locking a smart lock. Aloufi et al. (2019) proposed integrating the smart home security system with the web to interact with other security departments for the future Smart City, such as police for crime and civil defense for disasters. Hani et al. (2020) presented an efficient crime predictive system that could enable robust security management in a Smart Home Environment by identifying preventative procedures. Instead of gathering information from the crime scene after the crime, it can be stopped before happening by proper computing and quick action. Urquhart et al. (2022) highlights the importance of IoT devices sensing data interpretation and inference to shape police narratives during investigations.

6.2 PUBLICATIONS

- SBSEG 2021 (da Silva et al., 2021): publication of the article "Zero Trust Access Control with Context-Aware and Behavior-Based Continuous Authentication for Smart Homes", authors Giovanni Silva, Daniel Macedo, and Aldri Santos.

REFERENCES

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In Gellersen, H.-W., editor, *Handheld and Ubiquitous Computing*, pages 304–307, Berlin, Heidelberg. Springer Berlin Heidelberg.

Adams, C. (2005). *Impersonation Attack*, pages 286–286. Springer US, Boston, MA.

Ahmed, E. and Rehmani, M. H. (2017). Mobile edge computing: Opportunities, solutions, and challenges. *Future Generation Computer Systems*, 70:59–63.

Al-Hamadi, H. and Chen, I. R. (2017). Trust-based decision making for health iot systems. *IEEE Internet of Things Journal*, 4(5):1408–1419.

Al-Naji, F. H. and Zagrouba, R. (2020). A survey on continuous authentication methods in Internet of Things environment. *Computer Communications*, 163(August):109–133.

Al-rimy, B. A. S., Maarof, M. A., and Shaid, S. Z. M. (2018). Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers Security*, 74:144–166.

Alam, M. R., Reaz, M. B. I., and Ali, M. A. M. (2012). A review of smart homes—past, present, and future. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1190–1203.

Albrecht, K. and Mcintyre, L. (2015). Privacy nightmare: When baby monitors go bad [opinion]. *IEEE Technology and Society Magazine*, 34(3):14–19.

Almenárez, F., Marín, A., Campo, C., and García, R. C. (2005). TrustAC: Trust-based access control for pervasive devices. *Lecture Notes in Computer Science*, 3450(May):225–238.

Aloufi, K., Redwan, A., Abutarboush, Y., and Alharbi, A. (2019). Web based access control of smart home security system quranic mobile-learning system view project internet of things view project web based access control of smart home security system.

Alrajeh, N. A., Khan, S., and Shams, B. (2013). Intrusion detection systems in wireless sensor networks: A review. *International Journal of Distributed Sensor Networks*, 2013.

Alshammari, T., Alshammari, N., Sedky, M., and Howard, C. (2018). SIMADL: Simulated activities of daily living dataset. *Data*, 3(2):1–13.

Amraoui, N., Besrour, A., Ksantini, R., and Zouari, B. (2020). *Implicit and Continuous Authentication of Smart Home Users*, volume 926. Springer International Publishing.

Anthi, E., Williams, L., Słowińska, M., Theodorakopoulos, G., and Burnap, P. (2019). A supervised intrusion detection system for smart home iot devices. *IEEE Internet of Things Journal*, 6(5):9042–9053.

Ashibani, Y., Kauling, D., and Mahmoud, Q. (2019). Design and Implementation of a Contextual-Based Continuous Authentication Framework for Smart Homes. *Applied System Innovation*, 2(1):4.

Ashibani, Y. and Mahmoud, Q. H. (2019). User authentication for smart home networks based on mobile apps usage. *Proceedings of Communications and Networks, ICCCN*, 2019-July:1–6.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805.

Bace, R. G., Mell, P., et al. (2001). Intrusion detection systems.

Bakar, U. A. B. U. A., Ghayvat, H., Hasanm, S. F., and Mukhopadhyay, S. C. (2016). *Activity and Anomaly Detection in Smart Home: A Survey*, pages 191–220. Springer International Publishing, Cham.

Baran, P. (1964). On distributed communications networks. *IEEE Transactions on Communications Systems*, 12(1):1–9.

BBC (2020). Smart camera and baby monitor warning given by uk's cyber-defender. `https://www.bbc.com/news/technology-51706631`. Accessed in 16/07/2021.

Bell, D. E. LaPadula, L. J. (1973). Secure computer systems: Mathematical foundations. Technical report, MITRE CORP BEDFORD MA.

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., and Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161–180. Context Modelling, Reasoning and Management.

Biba, K. J. (1977). Integrity considerations for secure computer systems. Technical report, MITRE CORP BEDFORD MA.

Buck, C., Olenberger, C., Schweizer, A., Völter, F., and Eymann, T. (2021). Never Trust, Always Verify: A Multivocal Literature Review on Current Knowledge and Research Gaps of Zero-trust. *Computers Security*, page 102436.

Butun, I., Morgera, S. D., and Sankar, R. (2014). A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys Tutorials*, 16(1):266–282.

Butun, I., Osterberg, P., and Song, H. (2020). Security of the Internet of Things: Vulnerabilities, Attacks, and Countermeasures. *IEEE Communications Surveys and Tutorials*, 22(1):616–644.

Cardinaux, F., Brownsell, S., Hawley, M., and Bradley, D. (2008). Modelling of behavioural patterns for abnormality detection in the context of lifestyle reassurance. In *Iberoamerican Congress on Pattern Recognition*, pages 243–251. Springer.

Carlsen, J. (2021). Security school: Latest home security breaches and responses. `https://www.safewise.com/blog/latest-home-security-breaches-and-responses/#Recent_hacks_and_breaches`. Accessed in 16/07/2021.

Cebula, J. L. and Young, L. R. (2010). A taxonomy of operational cyber security risks. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

Chalhoub, G., Flechais, I., Nthala, N., Abu-Salma, R., and Tom, E. (2020). Factoring user experience into the security and privacy design of smart home devices: A case study. Association for Computing Machinery.

Chen, G. and Kotz, D. (2000). A survey of context-aware mobile computing research [Un estudio de la investigación sobre computación móvil sensible al contexto]. *Computer Science Technical Reports*, 1(2.1):1–16.

Chen, H., Finin, T., Joshi, A., Kagal, L., Perich, F., and Chakraborty, D. (2004). Intelligent agents meet the semantic web in smart spaces. *IEEE Internet Computing*, 8(6):69–79.

Chen, K., Zhang, S., Li, Z., Zhang, Y., Deng, Q., Ray, S., and Jin, Y. (2018). Internet-of-Things Security and Vulnerabilities: Taxonomy, Challenges, and Practice. *Journal of Hardware and Systems Security*, 2(2):97–110.

Chen, L., Hoey, J., Nugent, C. D., Cook, D. J., and Yu, Z. (2012). Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):790–808.

Choudhury, T., Philipose, M., Wyatt, D., and Lester, J. (2006). Towards activity databases: Using sensors and statistical models to summarize people's lives. *IEEE Data Eng. Bull.*, 29(1):49–58.

Cisco (2015). Cisco fog computing solutions: Unleash the power of the internet of things. `https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-solutions.pdf`. Accessed in 21/07/2021.

Consortium, N.-. (2021). ns-3 discrete-event network simulator for internet systems. `https://www.nsnam.org/`. Accessed in 19/10/2021.

Cook, D. J. (2007). Making sense of sensor data. *IEEE Pervasive Computing*, 6(2):105–108.

Cook, D. J., Crandall, A. S., Thomas, B. L., and Krishnan, N. C. (2012). Casas: A smart home in a box. *Computer*, 46(7):62–69.

CSA, C. S. A. (2023a). Building the foundation and future of the iot. `https://csa-iot.org/`. Accessed in 10/01/2023.

CSA, C. S. A. (2023b). connectedhomeip. `https://github.com/project-chip/connectedhomeip#readme`. Accessed in 10/01/2023.

CSA, C. S. A. (2023c). Matter: The foundation for connected things. `https://csa-iot.org/all-solutions/matter/`. Accessed in 10/01/2023.

Cui, Z., Zhao, Y., Li, C., Zuo, Q., and Zhang, H. (2019). An Adaptive Authentication Based on Reinforcement Learning. *2019 IEEE International Conference on Consumer Electronics - Taiwan, ICCE-TW 2019*, pages 2019–2020.

da Silva, G., Macedo, D., and dos Santos, A. (2021). Zero trust access control with context-aware and behavior-based continuous authentication for smart homes. In *Anais do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 43–56, Porto Alegre, RS, Brasil. SBC.

Darby, S. J. (2018). Smart technology in the home: time for more clarity. *Building Research & Information*, 46(1):140–147.

De Fuentes, J. M., Gonzalez-Manzano, L., and Ribagorda, A. (2018). Secure and usable user-in-a-context continuous authentication in smartphones leveraging non-assisted sensors. *Sensors*, 18(4).

de Matos, E., Amaral, L. A., and Hessel, F. (2017). Context-aware systems: Technologies and challenges in internet of everything environments. In *Beyond the Internet of Things*, pages 1–25. Springer.

Dennis, J. B. and Van Horn, E. C. (1966). Programming semantics for multiprogrammed computations. *Commun. ACM*, 9(3):143–155.

Devopedia (2020). Iot operating systems. `https://devopedia.org/iot-operating-systems`. Accessed in 29/08/2021.

Dey, A. (2001). Understanding and using context. *Personal and ubiquitous computing*, pages 4–7.

Dias, R. (2017). The 5 factors of authentication. `https://dojowithrenan.medium.com/the-5-factors-of-authentication-bcb79d354c13`. Accessed in 03/08/2021.

Dimitrakos, T., Dilshener, T., Kravtsov, A., La Marra, A., Martinelli, F., Rizos, A., Rosett, A., and Saracino, A. (2020). Trust aware continuous authorization for zero trust in consumer internet of things. *Proceedings - 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2020*, pages 1801–1812.

Doan, T. T., Safavi-Naini, R., Li, S., Avizheh, S., Muni Venkateswarlu, K., and Fong, P. W. (2018). Towards a resilient smart home. *IoT S and P 2018 - Proceedings of the 2018 Workshop on IoT Security and Privacy, Part of SIGCOMM 2018*, pages 15–21.

Domingo, J. S. (2023). You (probably) don't need gigabit internet. `https://www.nytimes.com/wirecutter/blog/what-is-gigabit-internet-do-you-need-it/`. Accessed in 18/06/2023.

Dong, M. and Ansari, N. (2020). Guest editorial: Special section on cyber-physical social systems—integrating human into computing. *IEEE Trans. on Emerging Topics in Computing*, 8(1):4–5.

Dutta, S., Chukkapalli, S. S. L., Sulgekar, M., Krithivasan, S., Das, P. K., and Joshi, A. (2020). Context Sensitive Access Control in Smart Home Environments. *Proceedings - 2020 IEEE 6th Intl Conference on Big Data Security on Cloud, BigDataSecurity 2020, 2020 IEEE Intl Conference on High Performance and Smart Computing, HPSC 2020 and 2020 IEEE Intl Conference on Intelligent Data and Security, IDS 2020*, pages 35–41.

Edwards, W. K. and Grinter, R. E. (2001). At home with ubiquitous computing: Seven challenges. In Abowd, G. D., Brumitt, B., and Shafer, S., editors, *Ubicomp 2001: Ubiquitous Computing*, pages 256–272, Berlin, Heidelberg. Springer Berlin Heidelberg.

Fan, X., Xie, Q., Li, X., Huang, H., Wang, J., Chen, S., Xie, C., and Chen, J. (2017). Activity recognition as a service for smart home: Ambient assisted living application via sensing home. In *2017 IEEE International Conference on AI Mobile Services (AIMS)*, pages 54–61.

Feng, H., Fawaz, K., and Shin, K. G. (2017). Continuous authentication for voice assistants. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, MobiCom '17, page 343–355, New York, NY, USA. Association for Comp. Machinery.

Gajewski, M., Batalla, J. M., Mastorakis, G., and Mavromoustakis, C. X. (2019). A distributed ids architecture model for smart home systems. *Cluster Computing*, 22:1739–1749.

Gazeta do Povo, C. (2019). Pela 2ª vez consecutiva, curitiba é finalista em prêmio mundial de cidades inteligentes. `https://www.gazetadopovo.com.br/haus/inovacao/curitiba-finalista-premio-mundial-cidades-inteligentes-2019/`. Accessed in 14/11/2019.

Geneiatakis, D., Kounelis, I., Neisse, R., Nai-Fovino, I., Steri, G., and Baldini, G. (2017). Security and privacy issues for an IoT based smart home. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2017 - Proceedings*, pages 1292–1297.

Georgeff, M. P. and Ingrand, F. (1989). Decision-Making in an Embedded Reasoning System. In *International Joint Conference on Artificial Intelligence*, Detroit, United States.

Ghansah, I. (2009). Smart grid cyber security potential threats. *Vulnerabilities and risks. California Energy Commission, PIER Energy-Related Environmental Research Program.*

Ghosh, N., Chandra, S., Sachidananda, V., and Elovici, Y. (2019). SoftAuthZ: A Context-Aware, Behavior-Based Authorization Framework for Home IoT. *IEEE Internet of Things Journal*, 6(6):10773–10785.

Google (2021). What sort of latency is acceptable? `https://developers.google.com/assistant/smarthome/faq#response-latency`. Accessed in 21/10/2021.

Gope, P., Gheraibia, Y., Kabir, S., and Sikdar, B. (2021). A secure iot-based modern healthcare system with fault-tolerant decision making process. *IEEE Journal of Biomedical and Health Informatics*, 25(3):862–873.

Guan, D., Yuan, W., Lee, S., and Lee, Y.-K. (2007). Context selection and reasoning in ubiquitous computing. In *The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007)*, pages 184–187.

Guus Schreiber, Y. R. (2014). Rdf 1.1 primer. `https://www.w3.org/TR/rdf11-primer/`. Accessed in 02/11/2022.

Hall, D. and Llinas, J. (1997). An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23.

Hani, M. D. J., Sheniha, M., Logeshwari, V., and Brindha, N. (2020). Iot-guard real-time security management in smart home environment. *International Research Journal of Engineering and Technology*.

Hassan, N., Gillani, S., Ahmed, E., Yaqoob, I., and Imran, M. (2018). The Role of Edge Computing in Internet of Things. *IEEE Communications Magazine*, 56(11):110–115.

Heartfield, R., Loukas, G., Budimir, S., Bezemskij, A., Fontaine, J. R., Filippoupolitis, A., and Roesch, E. (2018). A taxonomy of cyber-physical threats and impact in the smart home. *Computers and Security*, 78:398–428.

Heierman, E. O. and Cook, D. J. (2003). Improving home automation by discovering regularly occurring device usage patterns. In *Third IEEE International conference on data mining*, pages 537–540. IEEE.

Hitzler, P. (2021). A review of the semantic web field.

Hitzler, P., Krotzsch, M., and Rudolph, S. (2009a). *Foundations of semantic web technologies*. Chapman and Hall/CRC.

Hitzler, P., Parsia, B., Patel-schneider, P. F., and Rudolph, S. (2009b). OWL 2 Web Ontology Language Primer. *W3C Recommendation*, (December):1–123.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M., et al. (2004). Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21(79):1–31.

Hu, D. H. and Yang, Q. (2008). Cigar: Concurrent and interleaving goal and activity recognition. In *AAAI*, volume 8, pages 1363–1368.

Hu, V. C., Ferraiolo, D., Kuhn, R., Friedman, A. R., Lang, A. J., Cogdell, M. M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., et al. (2013). Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800(162):1–54.

Humayed, A., Lin, J., Li, F., and Luo, B. (2017). Cyber-Physical Systems Security - A Survey. *IEEE Internet of Things Journal*, 4(6):1802–1831.

Indulska, J., Sutton, P., Johnson, C., Montague, P., and Steketee, C. (2003). Context-Aware, Ambient, Pervasive and Ubiquitous Computing. *Computer Science - ACSW 2003*, 21(vi).

Intersoft Consulting, E. (2019). General data protection regulation, definitions. `https://gdpr-info.eu`. Accessed in 14/07/2021.

Jie, Y., Pei, J. Y., Jun, L., Yun, G., and Wei, X. (2013). Smart home system based on iot technologies. In *2013 International conference on computational and information sciences*, pages 1789–1791. IEEE.

Kalam, A., Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A., Saurel, C., and Trouessin, G. (2003). Organization based access control. In *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pages 120–131.

Kang, D.-J., Lee, J.-J., Kim, S.-J., and Park, J.-H. (2009). Analysis on cyber threats to scada systems. In *2009 Transmission & Distribution Conference & Exposition: Asia and Pacific*, pages 1–4. IEEE.

Kaspersky (2021). How safe are smart homes? `https://www.kaspersky.com/resource-center/threats/how-safe-is-your-smart-home`. Accessed in 01/10/2021.

Kayes, A. S., Kalaria, R., Sarker, I. H., Islam, M. S., Watters, P. A., Ng, A., Hammoudeh, M., Badsha, S., and Kumara, I. (2020). A survey of context-aware access control mechanisms for cloud and fog networks: Taxonomy and open research issues. *Sensors (Switzerland)*, 20(9).

Khan, S. and Loo, K.-K. (2009). Real-time cross-layer design for a large-scale flood detection and attack trace-back mechanism in ieee 802.11 wireless mesh networks. *Network Security*, 2009(5):9–16.

Khan, S., Loo, K.-K., and Din, Z. U. (2010). Framework for intrusion detection in ieee 802.11 wireless mesh networks. *Int. Arab J. Inf. Technol.*, 7(4):435–440.

Khan, W. Z., Ahmed, E., Hakak, S., Yaqoob, I., and Ahmed, A. (2019). Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235.

Khattak, H. A., Shah, M. A., Khan, S., Ali, I., and Imran, M. (2019). Perception layer security in Internet of Things. *Future Generation Computer Systems*, 100:144–164.

Kindervag, J. (2010). No more chewy centers: Introducing the zero trust model of information security.

Komninos, N., Philippou, E., and Pitsillides, A. (2014). Survey in smart grid and smart home security: Issues, challenges and countermeasures. *IEEE Communications Surveys and Tutorials*, 16(4):1933–1954.

Krombholz, K., Hobel, H., Huber, M., and Weippl, E. (2015). Advanced social engineering attacks. *Journal of Information Security and Applications*, 22:113–122. Special Issue on Security of Information and Networks.

Kuyucu, M. K., Bahtiyar, S., and Ince, G. (2019). Security and Privacy in the Smart Home: A Survey of Issues and Mitigation Strategies. *UBMK 2019 - Proceedings, 4th International Conference on Computer Science and Engineering*, pages 113–118.

Lal, N. A., Prasad, S., and Farik, M. (2015). A Review Of Authentication Methods. *International Journal of Scientific Technology Research*, 4(8):246–249.

Langley, D. J., van Doorn, J., Ng, I. C., Stieglitz, S., Lazovik, A., and Boonstra, A. (2021). The Internet of Everything: Smart things and their impact on business models. *Journal of Business Research*, 122(June 2018):853–863.

LBCA (2021). Lei geral de proteção de dados. `https://www.lgpdbrasil.com.br/`. Accessed in 25/07/2021.

Lee, H. (2020). Home IoT resistance: Extended privacy and vulnerability perspective. *Telematics and Informatics*, 49:101377.

Legrenzi, P., Girotto, V., and Johnson-Laird, P. (1993). Focussing in reasoning and decision making. *Cognition*, 49(1):37–66.

Liao, H. J., Richard Lin, C. H., Lin, Y. C., and Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24.

MarketsAndMarkets (2023). Smart home market size, share industry growth analysis report by product (lighting control, security access control, hvac control, smart speaker, smart kitchen and smart furniture), software and services, sales channel and region - global forecast to 2028. `https://www.marketsandmarkets.com/Market-Reports/smart-homes-and-assisted-living-advanced-technologie-and-global-market-121.html`. Accessed in 13/06/2023.

McQuigge, M. (2020). The quarantine act explained, as isolation becomes mandatory for some. `https://www.ctvnews.ca/health/coronavirus/the-quarantine-act-explained-as-isolation-becomes-mandatory-for-some-1.4868457`. Accessed in 06/01/2023.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing. `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf`. Accessed in 20/07/2021.

Milinković, A., Milinković, S., and Lazić, L. (2015). Choosing the right rtos for iot platform. *INFOTEH-JAHORINA*, 14:504–509. Cited By :5.

Miraoui, M. and El-Etriby, S. (2019). A context-aware authentication approach for smartphones. *2019 International Conference on Computer and Information Sciences, ICCIS 2019*, pages 1–5.

Mocrii, D., Chen, Y., and Musilek, P. (2018). IoT-based smart homes: A review of system architecture, software, communications, privacy and security. *Internet of Things*, 1-2:81–98.

Moreham, N. (2014). Beyond information: physical privacy in english law. *The Cambridge Law Journal*, 73(2):350–377.

Nicholson, A., Webber, S., Dyer, S., Patel, T., and Janicke, H. (2012). Scada security in the light of cyber-warfare. *Computers & Security*, 31(4):418–436.

Ning, Z., Guo, L., Rodrigues, J., and Obaidat, M. S. (2019). Guest editorial: Fog computing enabled internet of everything. *China Communications*, 16(3):iii–V.

Nurmi, P. and Floréen, P. (2004). Reasoning in Context-Aware Systems. *Position Paper. Department of Computer Science, University of Helsinki*, (1):1–6.

Ortiz, A. M., Hussein, D., Park, S., Han, S. N., and Crespi, N. (2014). The cluster between internet of things and social networks: Review and research challenges. *IEEE Internet of Things Journal*, 1(3):206–215.

Ouaddah, A., Mousannif, H., Abou Elkalam, A., and Ait Ouahman, A. (2017). Access control in the Internet of Things: Big challenges and new opportunities. *Computer Networks*, 112:237–262.

Padhi, P. K. and Charrua-Santos, F. (2021). 6g enabled industrial internet of everything: Towards a theoretical framework. *Applied System Innovation*, 4(1).

Pan, Z., Pacheco, J., Hariri, S., Chen, Y., and Liu, B. (2019). Context Aware Anomaly Behavior Analysis for Smart Home Systems. 13(5):261–274.

Panwar, N., Sharma, S., Mehrotra, S., Łukasz Krzywiecki, and Venkatasubramanian, N. (2019). Smart home survey on security and privacy.

Park, E., Cho, Y., Han, J., and Kwon, S. J. (2017). Comprehensive approaches to user acceptance of internet of things in a smart home environment. *IEEE Internet of Things Journal*, 4:2342–2350.

Park, J. and Sandhu, R. (2004). The uconabc usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174.

Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. *IEEE Communications Surveys Tutorials*, 16(1):414–454.

Pfleeger, C. and Pfleeger, S. (2016). Security in computing (4th edidtion, 2006).

Piccialli, F., Casolla, G., Cuomo, S., Giampaolo, F., and di Cola, V. S. (2020). Decision making in iot environment through unsupervised learning. *IEEE Intelligent Systems*, 35(1):27–35.

Pietschmann, S., Mitschick, A., Winkler, R., and Meißner, K. (2008). Croco: Ontology-based, cross-application context management. In *2008 Third International Workshop on Semantic Media Adaptation and Personalization*, pages 88–93.

Poppe, R. (2010). A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990.

Porambage, P., Ylianttila, M., Schmitt, C., Kumar, P., Gurtov, A., and Vasilakos, A. V. (2016). The quest for privacy in the internet of things. *IEEE Cloud Computing*, 3(2):36–45.

Quintal, K., Kantarci, B., Erol-Kantarci, M., Malton, A., and Walenstein, A. (2020). Enterprise Security with Adaptive Ensemble Learning on Cooperation and Interaction Patterns. *2020 IEEE 17th Annual Consumer Communications and Networking Conference, CCNC 2020*, pages 1–7.

Rahman, L. F., Ozcelebi, T., and Lukkien, J. (2018). Understanding iot systems: A life cycle approach. *Procedia Computer Science*, 130:1057–1062. The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops.

Rahmati, A., Fernandes, E., Eykholt, K., and Prakash, A. (2018). Tyche: A risk-based permission model for smart homes. *Proceedings - 2018 IEEE Cybersecurity Development Conference, SecDev 2018*, pages 29–36.

Reig, S., Carter, E. J., Kirabo, L., Fong, T., Steinfeld, A., and Forlizzi, J. (2021). Smart Home Agents and Devices of Today and Tomorrow : Surveying Use and Desires. pages 300–304.

Roche, C. (2003). Ontology: A Survey. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 36(22):187–192.

Ross, R. (2012). Guide for conducting risk assessments nist special publication 800-30 revision 1. *US Dept. Commerce, NIST, Gaithersburg, MD, USA, Tech. Rep.*

Sandhu, R. and Samarati, P. (1994). Access control: principle and practice. *IEEE Communications Magazine*, 32(9):40–48.

Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23.

Scott W. Rose, Oliver Borchert, Stuart Mitchell, S. C. (2020). Zero Trust Architecture - NIST Special Publication 800-207. *Nist*, page 49.

Setola, R. (2011). Cyber threats to scada systems. `http://panzieri.dia.uniroma3.it/MICIE/docs/Setola%20WS%20Micie%202011.pdf`. Accessed in 17/08/2021.

Shi, W. and Dustdar, S. (2016). The Promise of Edge Computing. *Ieee Computer Society*, (0018):17–20.

Shinkarenko, A. (2020). Internet of everything vs internet of things. `https://www.itransition.com/blog/internet-of-everything-vs-internet-of-things`. Accessed in 01/08/2021.

Sikder, A. K., Aksu, H., and Uluagac, A. S. (2020). A context-aware framework for detecting sensor-based threats on smart devices. *IEEE Trans. on Mobile Computing*, 19(2):245–261.

Sikder, A. K., Babun, L., Aksu, H., and Uluagac, A. S. (2019a). AEGIS: A Context-aware Security Framework for Smart Home Systems. *ACM Int. Conference Proceeding Series*, pages 28–41.

Sikder, A. K., Babun, L., Celik, Z. B., Acar, A., Aksu, H., McDaniel, P., Kirda, E., and Uluagac, A. S. (2019b). KRATOS: Multi-User Multi-Device-Aware Access Control System for the Smart Home. pages 1–12.

Soliman, M., Abiodun, T., Hamouda, T., Zhou, J., and Lung, C.-H. (2013). Smart home: Integrating internet of things with web services and cloud computing. In *2013 IEEE 5th international conference on cloud computing technology and science*, volume 2, pages 317–320. IEEE.

Stavroulakis, P. and Stamp, M. (2010). *Handbook of information and communication security*. Springer Science & Business Media.

Stoneburner, G., Goguen, A., Feringa, A., et al. (2002). Risk management guide for information technology systems. *Nist special publication*, 800(30):800–30.

Stouffer, K. A., Falco, J. A., and Scarfone, K. A. (2011). Sp 800-82. guide to industrial control systems (ics) security: Supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc).

Strang, T. and Linnhoff-Popien, C. (2004). A Context Modeling Survey. *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing*, Workshop o(4):1–8.

Sun, K., Zou, Y., Radesky, J., Brooks, C., and Schaub, F. (2021). Child safety in the smart home: Parents' perceptions, needs, and mitigation strategies. *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW2).

Tanenbaum, A., Wetherall, D., and Translations, O. (2011). *Computer Networks*. PRENTICE HALL BRASIL.

Urquhart, L., Miranda, D., and Podoletz, L. (2022). Policing the smart home: The internet of things as 'invisible witnesses'. *Information Polity*, 27:233–246.

US-CERT (2009). Cyber threat source descriptions. `https://us-cert.cisa.gov/ics/content/cyber-threat-source-descriptions`. Accessed in 17/08/2021.

Vanderbilt (2020). What is the difference between cps and iot? `https://blog.engineering.vanderbilt.edu/what-is-the-difference-between-cps-and-iot`. Accessed in 31/07/2021.

Wazzeh, M., Ould-Slimane, H., Talhi, C., Mourad, A., and Guizani, M. (2022). Warmup and transfer knowledge-based federated learning approach for iot continuous authentication.

Weiser, M. (1991). The computer for the 21 st century. *Scientific American*, 265(3):94–105.

Wireshark (2021). Pdml - packet description markup language. `https://gitlab.com/wireshark/wireshark/-/wikis/PDML`. Accessed in 24/08/2021.

Xing, Z., Pei, J., and Keogh, E. (2010). A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, 12(1):40–48.

Yang, X., Liu, J., Chen, Y., Guo, X., and Xie, Y. (2020). Mu-id: Multi-user identification through gaits using millimeter wave radios. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 2589–2598.

YE, J., COYLE, L., DOBSON, S., and NIXON, P. (2007). Ontology-based models in pervasive computing systems. *The Knowledge Engineering Review*, 22(4):315–347.

Yubico (2019). 2019 state of password and authentication security behaviors report. `https://pages.yubico.com/2019-password-and-authentication-report.html`. Accessed in 14/07/2021.

Zanero, S. (2017). Cyber-physical systems. *Computer*, 50(4):14–16.

Zeng, E., Mare, S., Roesner, F., Clara, S., Zeng, E., Mare, S., and Roesner, F. (2017). End User Security and Privacy Concerns with Smart Homes This paper is included in the Proceedings of the End User Security Privacy Concerns with Smart Homes. *Thirteenth Symposium on Usable Privacy and Security (SOUPS)*, (Soups):65–80.

Zhang, P., Liu, J. K., Richard Yu, F., Sookhak, M., Au, M. H., and Luo, X. (2018). A Survey on Access Control in Fog Computing. *IEEE Communications Magazine*, 56(2):144–149.

Zheng, Y.-L., Ding, X.-R., Poon, C. C. Y., Lo, B. P. L., Zhang, H., Zhou, X.-L., Yang, G.-Z., Zhao, N., and Zhang, Y.-T. (2014). Unobtrusive sensing and wearable devices for health informatics. *IEEE Transactions on Biomedical Engineering*, 61(5):1538–1554.

Zhou, J., Leppanen, T., Harjula, E., Ylianttila, M., Ojala, T., Yu, C., Jin, H., and Yang, L. T. (2013). Cloudthings: A common architecture for integrating the internet of things with cloud computing. In *Proceedings of the 2013 IEEE 17th international conference on computer supported cooperative work in design (CSCWD)*, pages 651–657. IEEE.

Zorzi, M., Gluhak, A., Lange, S., and Bassi, A. (2010). From today's intranet of things to a future internet of things: a wireless- and mobility-related view. *IEEE Wireless Communications*, 17(6):44–51.