Leveraging the Heterogeneity of the Internet of Things Devices to Improve the Security of Smart Environments



Klaudia Krawiecka

Keble College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2022

Acknowledgements

Institutional

Firstly, I would like to thank my supervisor, Professor Ivan Martinovic, for his continued support and guidance. I also thank Simon Eberz and Simon Birnbach for sharing their valuable knowledge with me.

In addition, I would like to express my gratitude to Jack Sturgess, who was not only a co-author of some of my research papers, but also supported me during challenging situations. Similarly, I would like to thank Ivo Sluganovic for his support. I am also extremely grateful to my assessors who have helped me shape this dissertation.

I want to acknowledge all members of our research group. In particular, I would like to thank Bushra AlAhmadi, Richard Baker, Freddy Barr-Smith, Martin Georgiev, Sebastian Köhler, Michał Piskozub, Yashovardhan Sharma, and Petra Stanković. Going through this DPhil journey would have been a much less enjoyable experience without you.

Special thanks to my CDT cohort for going through tons of essays and shared adventures. Especially, I want to acknowledge Selina Cho, Jack Kenny, Dennis Malliouris, Romy Minko, Mark Quinlan, Lonie Sebagh, Eva Stankova, and Tommaso de Zan.

I will be eternally grateful to Janet Sadler and Jordan Summers for their support and friendship. I also want to thank Katherine Fletcher, David Hobbs, and Maureen York for their hard work in assisting and supporting the students of our CDT program.

I am grateful to the EPSRC, Google, and $(ISC)^2$ for providing stipends that allowed me to continue the DPhil program.

Personal

I would like to thank my family for their deep faith in me and my abilities even when I did not believe in myself. I would like to thank my parents, Iwona and Maciej, for supporting and cheering me on every day. I also want to thank my brother, Karol, for supporting me with his positive attitude and sense of humor. I dedicate this doctoral dissertation to all of you.

I also want to thank Keble College and all the dear friends I met there, including Olmo Calzolari, Javi Pardo Díaz, Max Hoffer, Alun Vaughan Jackson, Gogulan Karunanithy, Bryony Robinson, Ștefan Săftescu, Edward Fleri Soler, Max Whitby, and Adam Zibak.

An important part of my DPhil journey included joining the Oxford Women in Computer Science Society (OxWoCS). Not only have I acquired many technical and leadership skills, but I have also met brilliant and inspiring women who have become my dearest friends. Special thanks to Paula Fiddi, Yuge Shi, and Catherine Vlasov for sharing this experience with me. Here, I also want to acknowledge the Perspektywy Education Foundation for many collaborations with OxWoCS throughout all these years.

I would also like to express my gratitude to my dear friends Alina Petrova and Evgeny Sherkhonov. Not only did you support me in all my battles, but I was always able to turn to you for advice whenever I needed it. Moreover, I will never forget your support when I got injured. I want you both to remember that you are a rare gem in this world.

Many thanks to my best friend Bogna Baliszewska. Even though you are on the other side of the globe, I have always felt your support, no matter the situation. My DPhil journey is coming to an end so I can finally visit you.

Lastly, I want to thank all my other friends for their continued support. Special thanks to David Egea Arrebola, Tasha Bharucha, Juli Brunhilde, Anirudh Ekambaranathan, Sylvia Gattas, Hayyu Imanda, Pranvera Kortoçi, Hawra Milani, Benedetta Mussati, Andrew Paverd, Matt Rogers, Claudine Tinsman, and my friends from the Wolfson College Boat Club.

Abstract

The growing number of devices that are being incorporated into the Internet of Things (IoT) environments leads to a wider presence of a variety of sensors, making these environments heterogeneous. However, the lack of standard input interfaces in such ecosystems poses a challenge in securing them. Among other existing vulnerabilities, the most prevalent are the lack of adequate access control mechanisms and the exploitation of cross-channel interactions between smart devices.

In order to tackle the first challenge, I propose a novel behavioral biometric system based on naturally occurring interactions with objects in smart environments. This system is designed to reduce the reliance on existing app-based authentication mechanisms of current smart home platforms and it leverages existing heterogeneous IoT devices to both identify and authenticate users without requiring any hardware modifications of existing smart home devices.

To be able to collect the data and evaluate this system, I introduce an end-to-end framework for remote experiments. Such experiments play an important role across multiple fields of studies, from medical science to engineering, as they allow for better representation of human participants and more realistic experimental environments, and ensure research continuity in exceptional circumstances, such as nationwide lockdowns. Yet cyber security has few standards for conducting experiments with human participants, let alone in a remote setting. This framework systematizes design and deployment practices while preserving realistic, reproducible data collection and the safety and privacy of participants.

Using this methodology, I conduct two experiments. The first one is a multi-user study taking place in six households composed of 25 participants. The second experiment involves 13 participants in a company environment and is used to study mimicry attacks on the biometric system proposed in this thesis. I demonstrate that this system can identify users in multi-user environments with an accuracy of at least 98% for a single object interaction without requiring any sensors on the object itself. I also show that it can provide seamless and unobtrusive authentication while remaining highly resistant to zero-effort, video, and in-person observation-based mimicry attacks. Even when at most 1% of the strongest type of mimicry attacks are successful, this system does not require the user to take out their phone to approve legitimate transactions in more than 80% of cases for a single interaction. This increases to 92% of transactions when interactions with more objects are considered.

To mitigate the second vulnerability, where an attacker exploits multiple heterogeneous devices in a chain such that each one triggers the next, I propose a novel approach that uses only dynamic analysis to examine such interactions in smart ecosystems. I use real-time device data to generate a knowledge graph that models the interactions between devices and enables the system to identify attack chains and vulnerable automations. I evaluate this approach in a smart home environment with 8 devices and 10 automations, with and without the presence of an active user. I demonstrate that such a system can accurately detect 10 cross-channel interactions that lead to 30 different cross-channel interaction chains in the unoccupied environment and 6 such interactions that result in 13 interaction chains in the occupied environment.

Contents

List of Figures **xi**

List of Abbreviations **xv**

1 Introduction **1**

1.1 Preface 1
1.2 Research Questions 5
1.3 Contributions 6
1.4 Outline 7

2 Background and Related Work **11**

2.1 Smart Environments 11
2.1.1 Device Capabilities 12
2.1.2 Challenges 14
2.1.3 Access Control 15
2.1.4 Cross-channel Interactions 17
2.1.5 Detection of Cross-channel Interactions 17
2.2 Biometric Systems 19
2.2.1 Single-biometric systems 19
2.2.2 Multi-biometric systems 20
2.2.3 Multi-sensor fusion 20
2.2.4 Interaction-based biometric systems 20
2.2.5 Mimicry Attacks 21
2.2.6 Metrics 22
2.3 Remote Experimentation 23
2.3.1 Remote Experimentation Practices 23
2.4 Research Scope 24
2.4.1 Research Gaps 24
2.4.2 Adversary Model 25

3 Preliminary Study **27**
3.1 Proving Co-location . . . 27
3.2 Scope . . . 30
3.3 Protocol Overview . . . 31
3.4 System and Adversary Model . . . 31
3.5 System Prototype . . . 34
3.6 Deployment . . . 38
3.7 Evaluation . . . 39
3.7.1 Security Analysis . . . 39
3.7.2 Entropy of User Interactions . . . 41
3.7.3 Recurring sequences . . . 42
3.7.4 AR Performance . . . 43
3.8 Summary . . . 44
3.9 Lessons Learned . . . 45

4 Plug-and-play: Framework for Remote Experimentation **47**
4.1 Motivation . . . 47
4.2 Related Work . . . 48
4.2.1 Controlled Field Studies . . . 49
4.3 Overview of the Remote Experimentation Framework . . . 50
4.3.1 Goals and Requirements . . . 50
4.3.2 Framework Architecture . . . 52
4.4 Case Study . . . 59
4.4.1 Architecture . . . 61
4.4.2 Survey Results . . . 65
4.5 Discussion . . . 67
4.5.1 Research Ethics . . . 67
4.5.2 Lessons Learned . . . 70

5 `BeeHIVE`: Behavioral Biometric System based on Object Interactions in Smart Environments **73**
5.1 Motivation . . . 74
5.1.1 Deployment Scenarios . . . 76
5.2 Related Work . . . 76
5.3 System Design . . . 77
5.3.1 Design objectives . . . 78
5.3.2 System Model . . . 79
5.3.3 Adversary Model . . . 80
5.4 Experimental Design . . . 81
5.4.1 Experiment 1 . . . 82

5.4.2 Experiment 2 83
5.5 Methods 85
5.5.1 Preprocessing 86
5.5.2 Feature extraction 87
5.5.3 Feature selection 88
5.5.4 Multi-sensor fusion 89
5.5.5 Training 91
5.5.6 Classification tasks 92
5.6 System Evaluation 92
5.6.1 Identification performance 92
5.6.2 Authentication performance 95
5.6.3 Distinctiveness of sensor features 103
5.7 Discussion 105

6 Butterfly Effect: Finding Cross-Channel Interactions in Smart Ecosystems using Dynamic Analysis 107
6.1 Motivation 108
6.1.1 Challenges 109
6.2 Related Work 110
6.2.1 Dynamic Analysis of Device Interactions 110
6.3 System Design 111
6.3.1 System Model 111
6.3.2 Adversary Model 112
6.4 Experimental Design 114
6.4.1 Testbed Overview 114
6.4.2 Data Collection 115
6.4.3 Experiment in an Unoccupied Environment 115
6.4.4 Experiment in an Occupied Environment 116
6.5 Methods 116
6.5.1 Causal Analysis 117
6.5.2 Knowledge Graphs 119
6.5.3 Ontology 119
6.5.4 Pre-processing 120
6.5.5 Graph Construction 121
6.5.6 Identifying Cross-channel Interaction Chains 121
6.5.7 Classification Metrics 122
6.6 Evaluation 122
6.6.1 Causal Analysis 123
6.6.2 Cross-channel Interaction Chains 132
6.7 Discussion 136

7 Conclusions **139**
7.1 Summary 139
7.2 Impact 142
7.3 Future Work 144

Appendices

A Instructions for User Study **149**
A.1 `BeeHIVE` – Information 149
A.2 `BeeHIVE` – User Manual 150

B House Layouts in User Study **153**

References **157**

List of Figures

1.1 The structure of this thesis. 8

2.1 Two examples of different Internet of Things deployments. Users can either buy smart devices from various vendors (a) or they can make their smart environment components (b). 13

3.1 **Location-based establishment of secure connection.** Using the insecure channel, the devices first exchange their public keys and the public information about the connected devices available for input and output. Next, the smart hub SH generates a commitment with nonces K_1 and K_2 and input lists, after which the user performs a set of input actions based on AR guidance. Finally, the gateway opens the second part of the commitment and performs output actions to prove its authenticity by showing the ability to control the connected devices. Finally, user U_A verifies and confirms/aborts the execution in the last message. 32

3.2 **Overview of the system model presented in this thesis**. After the user U_A requests remote access to smart devices D_1-D_3 from the gateway SH, the gateway asks U_A to prove that he has physical access to D_1-D_3 by instructing the user to interact with these devices (1). U_A uses an out-of-band physical channel (2) to interact with the connected devices. Next, SH uses an authenticated channel to communicate with connected devices to very the occurrence of the requested interactions (3). If the verification has been successful, SH grants remote access to U_A (4). U_A's mobile phone communicates with the gateway via an unauthenticated channel. The adversary is located outside the smart environment; however, he can observe the interior of this environment. 33

3.3 **Overview of the user interface.** In order to authenticate the exchanged keys, the mobile application indicates the tasks that the user has to perform to prove their physical presence in the smart environment, as well as the actions that the smart hub will perform to authenticate. . . . 36

4.1 The four phases of the proposed framework. Phases 2 and 3 are carried out by external entities such as the institutional review board and experiment participants. 50

4.2 Review of sex and age distribution of the participants in my remote experiment. 60
4.3 Raspberry Pis (RPIs) mimicked smart devices. The participants could freely arrange these in the room. Each RPI board contains an IMU sensor that measures acceleration and has a built-in gyroscope and magnetometer. Each board is equipped with a microphone and a contact sensor. 62
4.4 Participants placed sensor boards on various household objects, including an oven door, a microwave, a drawer, and a coffee maker. These are examples of deployments from two households that participated in my remote experiment. 63
4.5 An overview of communication between different nodes during the remote experiment. The MQTT broker that relays messages between the mobile application and sensor devices is configured on the server that is placed in a research lab. The backup of the data is stored on the same server. The mobile device and sensors reside in the participants' environments. 63
4.6 The left picture shows the main screen of the controller, where the participant enters his name and the number of deployed sensor nodes. The participant is then redirected to the second screen, which displays the current trial number and the start button. Before this button appears, the application checks whether the sensor nodes are ready for data collection. 64

5.1 An overview of the `BeeHIVE` system. As the user interacts with the printer, sensors embedded in smart objects surrounding the user and the printer record these interactions. Physical signals generated from the user's movements are picked up by sensors such as accelerometers, pressure sensors, and microphones, and are used to profile them. The system authenticates the user before allowing them to perform certain actions, such as payments. 78
5.2 A simplified layout of the room and the arrangement of the objects $O1 - O8$ the participants interacted with during the experiment. 84
5.3 The diagram provides an overview of the processing pipeline of a multi-sensor fusion system. The system extracts relevant features from U_L's interactions with objects O_1 to O_n and supplies them to their base classifiers. Then, the first-level predictions P_1 to P_n are fed into a meta-classifier (*i.e.*, a voting or stacking classifier) that computes the final prediction P_F. 85

5.4 As users interact with smart devices, signals from on-device sensors are collected and processed by the system. Signals (a) and (b) are generated by the participant U_1 that has interacted with the smart object $O4$. $O2$ picked up additional input from the same interaction with object $O4$ as they were co-located. More specifically, these plots illustrate the movements on the x axis of the gyroscope (a) of $O4$ and RMS values (b) picked up by the microphone of $O2$. Red dotted lines indicate the start (t_0) and end (t_1) of the T task while the green dashed lines denote $\pm$ 1-second windows. 86

5.5 Averaged accuracy scores of different ensembles of unique objects in $H6$ for the voting meta-classifier and the COMBINED configuration. Each such ensemble is trained and tested separately, then the scores are averaged across the ensembles of the same type (*e.g.*, pairs, triples of unique objects). The 95% confidence intervals are given for each ensemble. The y-axis cutoff range is between 0.75 and 1.00. 96

5.6 The plots above show the ROC curves for three system configurations respectively based on average FARs from single interaction types. Each curve represents a different group of attacks, *i.e.*, zero-effort (blue), in-person (green), and video-based (yellow) attacks. 98

5.7 Averaged False Reject Rates (FRRs) at different False Acceptance Rates (FARs) thresholds calculated based on the performance of different ensembles of unique objects for two meta-classifiers and the OFF-OBJECT configuration. Each such ensemble is trained and tested separately, then the scores are averaged across the ensembles of the same type (*e.g.*, pairs, triples of unique objects). 101

6.1 An overview of the system. First, the system extracts time-series signals from smart devices (1). These signals are then analysed to find triggers—*i.e.*, relevant sources R, for a target T (2). Based on the outcome of this analysis and after the pre-processing of the data (3), the knowledge graph is built (4), and the system flags potential cross-channel violations to users (5). 111

6.2 The diagram presents my threat model. The adversary (Adv) compromises a vulnerable device (in red) to target other devices (in green). This leads to triggering automation that opens the window, allowing Adv to enter the house. 112

6.3 This diagram provides a representation of the room in which I conducted the experiments, indicating device placement. 114

6.4 The diagram provides a general overview of the schema of the knowledge graph. Two types of triple exist in this schema. The first one consists of a device and capability pair along with an edge representing a directional relationship between the two nodes. The second one involves the capability-to-capability relationship with a lag (l_x) added as a graph edge parameter. 120
6.5 Comparison of knowledge graphs built based on the causal analysis of cross-channel interactions between smart devices during observations of different duration. False positives are depicted with red dashed arrows. The first part shows the graphs formed in the unoccupied IoT environment (a–d) in four time intervals of 15, 30, 45, and 60 minutes. *(cont.)* 124
6.5 *(cont.)* The second part presents the graphs constructed in the occupied environment (e–h) in four time intervals of 30, 60, 120, and 240 minutes. 125

B.1 The household layouts in the user study. 155

List of Abbreviations

ACC . . . . . Accelerometer.

AR . . . . . Augmented reality.

DDoS . . . . . Distributed Denial of Service.

FAR . . . . . False acceptance rate.

FP . . . . . False positive.

FRR . . . . . False rejection rate.

GPS . . . . . Global positioning system.

GYRO . . . . . Gyroscope.

HAR . . . . . Human activity recognition.

IFTTT . . . . . If This Then That, a paradigm for Internet of Things automations.

IMU . . . . . Inertial measurement unit.

IoT . . . . . Internet of things.

MAG . . . . . Magnetometer.

MIC . . . . . Microphone.

MITM . . . . . Man-in-the-middle.

ML . . . . . Machine Learning.

MQTT . . . . . MQ Telemetry Transport, a publish-subscribe network protocol.

NFC . . . . . Near-field communication.

PIN . . . . . Personal identification number.

PIR . . . . . Passive infrared.

RFID . . . . . Radio frequency identification.

RMI . . . . . Relative mutual information.

RPI . . . . . Raspberry Pi, a single-board computer.

RSS . . . . . Received signal strength.

SPL . . . . . Sound pressure level.

TP . . . . . True positive.

VPN . . . . . Virtual private network.

SDK . . . . . Software development kit.

1 Introduction

Contents

1.1 **Preface** 1
1.2 **Research Questions** 5
1.3 **Contributions** 6
1.4 **Outline** 7

1.1 Preface

During the 1990s, there was a widespread anticipation among researchers and industry professionals that the Internet of Things (IoT) would soon become an integral part of our daily lives. Kevin Ashton, supported by other researchers from the Massachusetts Institute of Technology, not only coined the term "Internet of Things" but also introduced the radio frequency identification (RFID) standard, which played a significant role in boosting the growth of the IoT market. The earliest of such inventions included an internet-controlled toaster and a camera [1]. In general, the IoT allowed for not only remote control of electronic devices by humans but also communication and interaction between objects within a shared context in the same environment.

The collaborative and heterogeneous nature of smart devices contributed to the emergence of **smart environments**. As manufacturers continued to make smart objects

more compact, their popularity and market demand surged. Initially, users owned individual objects such as smart watches, air quality measuring devices, or smart cameras, but today, fully automated smart homes, offices, or factories are commonplace. Key elements of such infrastructures are the diverse sensors that capture information from the physical world and convert it into digital signals that can be interpreted by other machines.

A significant benefit of such environments is that even though some devices may have limited capabilities because of the absence of suitable input and output interfaces, they can still be integrated into a larger infrastructure and complement each other. The primary objective of IoT components is to gather and analyze diverse data types to enhance daily activities and enrich user experiences. Nonetheless, this kind of data can be readily abused. For example, among the aggregated data, we can find information about one's health, finances, and daily routine. Obtaining such detailed information about the user's routine may violate not only their privacy but also safety [2]. Even seemingly trivial details, such as data on energy usage in a smart home, can be utilized to monitor the user's activities inside the house [3], leaving them vulnerable to burglars who could exploit the information to determine when no one is home.

While dynamic IoT infrastructures have the potential to also optimize industrial processes [4], their deployment can pose significant risks to businesses and national security if compromised. The reliable and seamless operation of production machines is crucial to ensure the quality of manufactured products. For example, even the slightest malfunction in machines that produce vehicle parts can have catastrophic consequences for consumers. As a result, it is essential to ensure that smart infrastructures incorporate proper access controls, and that adequate security measures are put in place.

More generally, access to any type of sensitive operation that can occur in a smart environment, must be limited to authorized users only. For example, smart windows in a child's bedroom should not open unless the parent is present, and a child should not be able to order multiple candy bars using a smart refrigerator. Similarly, not all office workers should have access to the history of a smart printer, and visitors in a guesthouse should not be able to alter credentials on smart devices that do not belong to them.

This can be mitigated by requiring the user to initiate or approve every operation through a companion application running on their smartphone. However, this negatively affects usability as it requires users to interact with their phone for every operation. Recent research focuses on leveraging sensing capabilities of smart devices to provide a seamless access control. Built-in sensors have been utilized to capture various types of contextual data to authenticate users, including voice, breath, or gait [5–8]. However, such systems are vulnerable to mimicry attacks where an adversary impersonates the legitimate user. Therefore, several other approaches have been proposed that rely on combinations of different inputs [9–14].

Although these approaches are promising, they do not always utilize the full potential of co-located heterogeneous devices in smart environments. Thus, in this thesis, I propose and evaluate a novel biometric system to facilitate access control in smart environments.

Collecting data in an environment that closely mimics a user's natural environment is crucial when studying a new biometric. Remote experiments with human participants are a valuable tool in various other fields, including medical science and engineering, as they offer a more realistic experimental environment and better representation of human participants, while ensuring research continuity in exceptional circumstances. Cyber security, however, has few standards for conducting such experiments. As a result, in preparation for studying the new biometric, I have developed an end-to-end framework for remote experimentation. This framework serves to standardize the design and deployment practices while simultaneously preserving realistic, reproducible data collection and safeguarding the safety and privacy of all participants involved. My proposed framework can be used in a wide range of studies that aim to capture users' natural behavior in certain contexts. One particular area of application is biometric studies, where this framework can help improve the real-world applicability of the proposed systems.

Although the absence of suitable access control mechanisms presents a major obstacle to safeguarding smart environments, there exist additional security concerns. A newly developing area of research is focused on the potential risks that may result from the interaction between smart objects across different communication channels. While the heterogeneity of IoT devices can serve as a means to safeguard smart ecosystems, it can

also expose them to vulnerabilities. Smart devices frequently produce new events and share data in ways that are not immediately evident and hard to predict.

Expanding the range of automations and devices within a smart environment enhances its capabilities. Each smart device comes equipped with a distinct set of sensors and is situated in a specific location, providing it with unique functions within the network of connected devices. The on-device sensors determine the communication channels available to the IoT device, while the device's location affects its range of communication.

The majority of communication protocols are based on the trigger-condition-action (TCA) model, as it is easily comprehensible for humans. Companion apps or smart hubs feature user-friendly interfaces that allow users to manage and specify the interactions between devices. However, these interfaces only provide a limited depiction of the actual communication model within the ecosystem. This is because physical signals can traverse different channels and be detected by various sensors. For instance, when a user enters a room, this event may be detected by microphones, accelerometers, and temperature sensors. In other words, different devices may capture the same event or physical interaction via diverse modalities. Moreover, the operation of one device can have a direct or indirect impact on another, which may result in a chain reaction in some cases. A prime example of such a chain reaction is when a user sets up (1) a smart heater to turn on when someone enters the room, (2) a washing machine to turn on one hour prior to the user's arrival from work, and (3) a smart window to open when the temperature surpasses a certain threshold. This could result in an undesirable cascade of events, where the vibrations caused by the washing machine might trigger the heater, which in turn may cause the smart window to open due to the temperature increase.

The preceding example is one of many instances in which unintended device interactions can impact the entire smart environment's operation. Therefore, in this thesis, apart from aforementioned contributions, I also propose a knowledge-graph-based modeling approach that leverages run-time dynamic analysis to extract and categorize information regarding direct and indirect signal propagation channels in smart environments. Previous research in this domain has focused on using static analysis of companion apps to model cross-channel interactions, but these approaches overlook a crucial factor: the physical

location of the devices [15–17]. This information can be exploited by malicious actors to expose interaction-based vulnerabilities in smart environments. On the other hand, it can also assist users in detecting unintentional device interactions and protecting their IoT ecosystems.

In summary, this thesis delves into ways of mitigating various attack vectors in smart environments. In particular, I present a novel biometric that can provide contextual information about users to aid in authentication within such environments. Traditional biometric systems are often compromised through mimicry attacks, hence I propose a more resilient approach to address this issue. Furthermore, I create and implement a dynamic modeling technique that detects cross-channel interactions between smart devices, thereby preventing malicious actors from exploiting the smart ecosystem for their own purposes.

1.2 Research Questions

In this thesis, I focus on answering the following questions:

1. **How can the data collection process be improved to study the natural behavior of users in smart environments?** In my work, I research methods that facilitate the study of biometric systems in users' natural environments. In addition, through this work, I explore how a remote experiment can support the recruitment of participants who would otherwise not be included in the study due to location, mobility issues, or external factors such as global pandemics.

2. **How can physical signals generated during user interactions with heterogeneous smart devices be used for access control?** This thesis explores whether heterogeneous smart devices can be used to improve the security of smart environments by providing unique signatures of users' interactions. It explores how physical signals that result from such interactions can be used to facilitate the identification and authentication of the user.

3. **How can dynamic analysis of physical signals generated by smart devices help model cross-channel interactions in smart environments?** My thesis also examines whether the data captured by heterogeneous sensors installed on the device can be used to model cross-channel interactions between smart devices in smart environments. Compared to the existing solutions, I focus only on dynamic analysis, because I consider the assumptions made in previous works to be unrealistic considering modern smart environments.

1.3 Contributions

To address my research questions, I begin by examining user interactions with smart devices in a preliminary study and constructing a pairing protocol to evaluate the feasibility of using a distributed IoT infrastructure to secure smart environments. As I recognize the unique nature of these interactions and their potential to serve as a foundation for a novel biometric authentication system, I establish a robust experimentation framework for gathering realistic data samples from human participants. Subsequently, I devise and deploy an interaction-based biometric system to evaluate the practicality of applying this approach for the purposes of identification and authentication. Finally, I build a system that derives the cross-channel interaction model between smart devices based on the causal dependencies between low-level physical sensor signals.

To sum up, the main contributions of this dissertation are:

1. In order to provide an accurate insight into the behavior of smart environment users as well as to study mimicry attacks, I propose a novel remote experimentation framework to collect data in users' natural environments. This work is motivated by the fact that traditional experiments are carried out in artificial environments set up in research facilities, influencing user behavior and yielding less realistic data. By comparison, the data obtained from sensors *in situ*, with their background noises and genuine agent behavior, are more representative of the natural behavior and habits of users. This research work has been published and presented at the European Symposium on Usable Security [18].

2. To facilitate access control in smart environments, I explore the use of heterogeneous sensor data from users' daily interactions with physical objects to implicitly identify and authenticate users without having to change smart environment hardware configuration or adjust users' behavior. The first part of this work discusses the identification performance of the proposed system and it has been published at the IEEE workshop on the Internet of Safe Things co-located with the 43rd IEEE Symposium on Security and Privacy [19]. The second part of this work has been submitted to the 20th International Conference on Security and Cryptography (SECRYPT).

3. Finally, to detect potential security and safety violations in smart environments, I propose a novel cross-channel detection system that uses dynamic analysis to build a knowledge graph that models interactions between heterogeneous channels of smart devices. This work has been submitted to the 20th The International Conference on Security and Cryptography (SECRYPT).

Although I am the lead author of all the aforementioned publications, my co-authors, including Jack Sturgess, Simon Birnbach, Simon Eberz, and Alina Petrova, assisted me with conducting the experiments and the preparation of the research papers. Furthermore, the findings from my preliminary research informed the development of the MoveAR system, which utilizes user interactions with 3D objects to enable seamless authentication for AR headsets [20]. I am the third author of this work, which has been published at the Cyber-Physical System Security workshop co-located with the 16th ACM ASIA Conference on Computer and Communications Security.

1.4 Outline

Figure1.1 presents the outline of this thesis. The dotted arrows show how different chapters impact each other. The green blocks show the main contributions discussed in this dissertation. The blue block indicates supporting research, namely the preliminary work which outcomes I used in other chapters. The thesis is structured as follows:

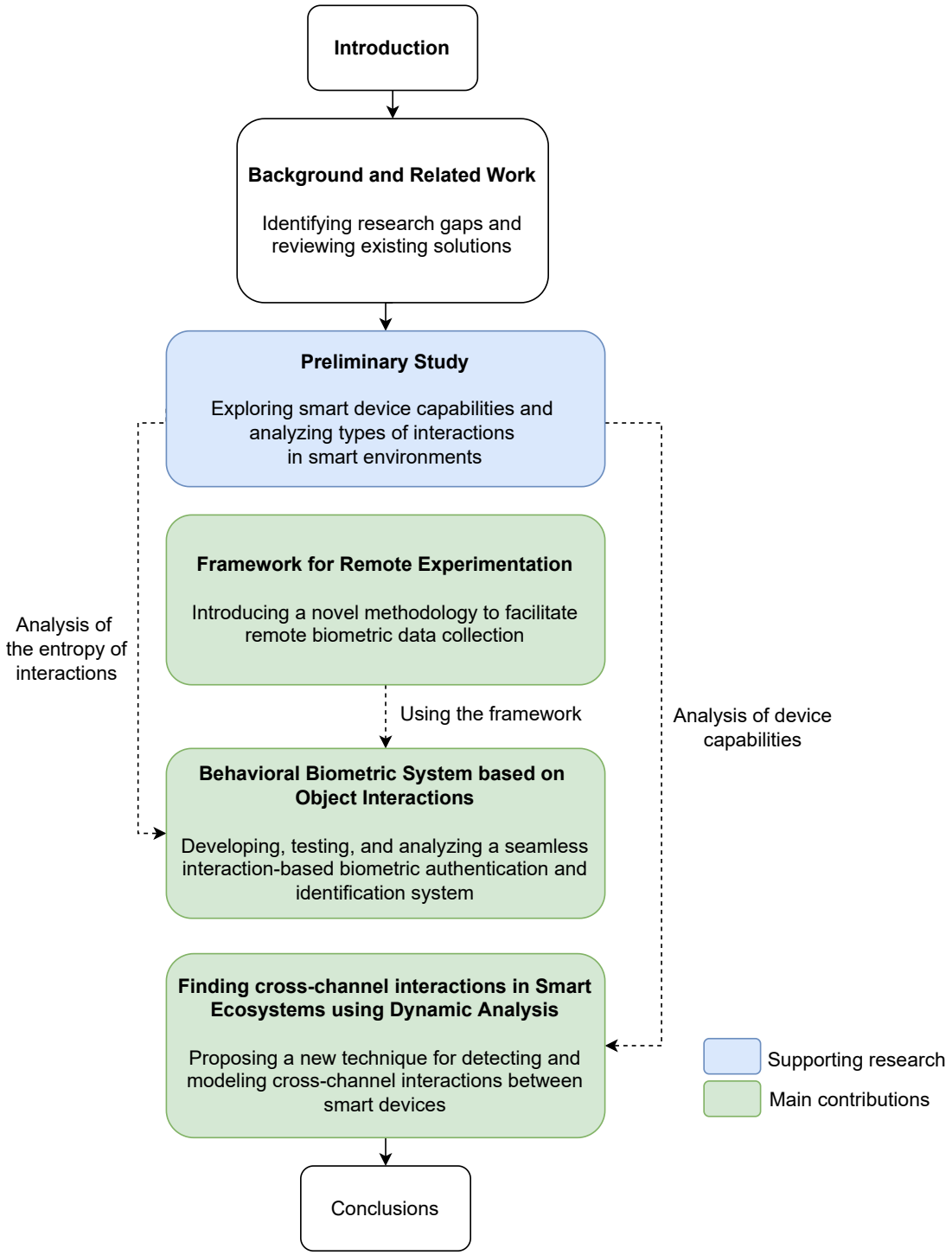


Figure 1.1: The structure of this thesis.

Ch1. This chapter describes the motivation and lists the main research contributions presented in this dissertation.

Ch2. The second chapter provides an overview of modern smart environments and their components and describes current challenges in securing such ecosystems. In addition, it discusses the state-of-the-art solutions to tackle these challenges. This chapter also presents metrics for evaluating these solutions, as well as the methods of data collection and experimentation in smart environments.

Ch3. This chapter presents the results of the preliminary research analysis that inspired subsequent projects and helped to formulate research questions for this thesis. It evaluates the proposed access control mechanism and describes its shortcomings on the basis of which the following chapters improve on.

Ch4. The fourth chapter defines a proposed framework for remote experiments in smart ecosystems. It discusses its components, and case study, and includes a discussion on research ethics. This methodology is used to design and conduct experiments for the other research projects described in this work.

Ch5. This chapter describes a proposed interaction-based access control mechanism that leverages the heterogeneity of smart devices to identify and authenticate users in smart environments. The two experiments described in this chapter use the previously mentioned framework for remote experimentation. The system presented in this chapter is compared to existing solutions and evaluated using the described metrics.

Ch6. The sixth chapter outlines a novel, dynamic approach to detecting cross-channel interactions between smart devices in smart ecosystems. While previous chapters focused on the study of how heterogeneous sensors can be used to capture physical signals resulting from interactions between devices and users, in this chapter I focus on interactions between various devices that can lead to potential violations in smart ecosystems. This solution is evaluated through two experiments in a smart environment with and without an active user present.

Ch7. This chapter summarizes the insights and key conclusions from the research work presented in this dissertation. It also describes future work that can be done to extend this work.

2

Background and Related Work

Contents

2.1 Smart Environments 11
2.1.1 Device Capabilities 12
2.1.2 Challenges 14
2.1.3 Access Control 15
2.1.4 Cross-channel Interactions 17
2.1.5 Detection of Cross-channel Interactions 17
2.2 Biometric Systems 19
2.2.1 Single-biometric systems 19
2.2.2 Multi-biometric systems 20
2.2.3 Multi-sensor fusion 20
2.2.4 Interaction-based biometric systems 20
2.2.5 Mimicry Attacks 21
2.2.6 Metrics 22
2.3 Remote Experimentation 23
2.3.1 Remote Experimentation Practices 23
2.4 Research Scope 24
2.4.1 Research Gaps 24
2.4.2 Adversary Model 25

2.1 Smart Environments

Nowadays, many everyday devices and appliances are enriched with sensors of various types, which contribute to the automation of tasks that have been manually performed

by users in the past. These heterogeneous objects are called `smart devices`. Such devices have a new ability to understand the surrounding environment and can adapt their operation to its changing conditions. The network traffic generated by smart devices is diverse, variable, and also characterized by a large volume [21]. This diversity has originated from different data formats, I/O types, and semantics, which in turn are determined by different vendors, the most popular of which is Samsung SmartThings [22]. Typically, the smart devices are centrally managed through a `smart hub` or a mobile application, which is commonly referred to as a `companion app`. The smart hub can be a physical device to which smart devices are connected using various communication protocols, including MQ Telemetry Transport (MQTT), Bluetooth, Zigbee, Z-Wave, or a cloud-based endpoint (*e.g.*, Azure IoT Hub) [23]. This central infrastructure that connects all devices allows the users to configure `automations`. The automations are expressed according to the trigger-condition-action (TCA) model because this model is easily understood by humans [24, 25]. The companion apps or smart hubs provide intuitive interfaces that allow users to control and specify the interactions between smart devices. For example, the user can set up the lights to turn on (action) when they enter the house (trigger) and the heater to start heating the room (action) if the temperature drops below a certain level (trigger). The trigger can be expressed as property such as time, device state, user geolocation, or weather conditions, while the action set includes changing the device state, sending notifications to users, or changing location. All aforementioned elements together create one ecosystem – a `smart environment`.

2.1.1 Device Capabilities

Due to the variety of built-in sensors and actuators, smart devices possess many different capabilities [26]. For example, the smart air quality monitor can read temperature, pressure, humidity, and carbon dioxide concentration in the air. For comparison, the multipurpose smart device not only reads the temperature and humidity but also detects movement. Therefore, these smart devices have two common channels, namely, both can detect changes in humidity and temperature. However, they also have other modalities



(a) Smart air quality meter by Awair equipped with humidity, temperature, and carbon dioxide sensors.

(b) Custom-made Raspberry Pi smart device with WiFi module and push button.

Figure 2.1: Two examples of different Internet of Things deployments. Users can either buy smart devices from various vendors (a) or they can make their smart environment components (b).

that, even though they are not shared through the same channels, can allow both devices to detect similar events.

According to the survey conducted by Zeng *et al.* [22], among the most common consumer smart devices are smart switches, motion sensors, cameras, lights, and thermostats. Some participants of the survey also mentioned using RaspberryPi (RPI) boards to build custom devices. Figure 2.1 shows two possible deployments of smart devices. Despite the varied input and output capabilities of such devices, many are equipped with a similar set of sensors that includes accelerometers, gyroscopes, and magnetometers or a combination of these embedded in an inertial measurement unit (IMU). IMUs are commonly used as an anti-tampering mechanism while only accelerometers are utilized to measure the level of vibrations during the operation of the device.

As mentioned, whether the devices have a similar set of sensors or not, they can detect the same physical context [27]. For example, if a user has entered a room, one device can detect its movement and the other may register a change in the concentration of carbon dioxide in the air. This is possible not only because of the sensor set but also because of the physical location of these devices. If they are co-located, *i.e.*, they are in proximity to one another, they can detect physical signals originating from the same event. This has been used, *inter alia*, to improve existing access control mechanisms for smart environments.

2.1.2 Challenges

Smart environments have been broadly classified into two categories, personal and enterprise [21]. Despite various similarities, these categories represent different contexts and characteristics, leading to distinct threat models and security challenges. In this work, I focus on smart homes and offices, because these environments are currently the most popular. Users of such environments can be both children and the elderly, who exhibit significantly different skill sets [28]. As the main goal of the smart device vendors is to make smart environments accessible to all [29], any system to be deployed in such an environment must follow the same principle. Moreover, for the same reason, any such system should be tested by a diverse group of participants. Multi-user environments face many challenges, not only because of different user capabilities, but also because of factors such as power imbalance that can lead to privacy violations and, at worst, abuse [30]. Apart from the issues related to access control among members of the same household, there are more threats and they concern every element of smart infrastructure.

Because such environments are complex, there is plenty of attack vectors [31]. Since only a few common standards or guidelines for the construction of smart devices exist, manufacturers are often not focusing on securing such devices. Also, it is not their priority, especially considering the low cost of such devices. Typical attack vectors on smart devices include hard-coded credentials that are shared between devices and can be easily guessed, insecure factory settings, or vulnerable services [32]. On the other hand, the companion apps can be overprivileged, or allow for exploitation of the smart devices or abuse of automations. Moreover, such applications may lack encryption or implement vulnerable communication protocols that enable man-in-the-middle (MITM) attacks. The smart hubs can suffer from similar issues, including running vulnerable software, and weak credentials.

In fact, misconfigured devices and weak passwords are among the major contributors to resource exploitation in smart environments. For example, Bashlite and its successor, the Mirai malware, used misconfigured networks to transform multiple electronic devices into botnets [33, 34]. These botnets have been used to launch Distributed Denial of Service (DDoS) attacks around the world. Bashlite and Mirai are not the only examples

of using the dynamic infrastructure of the Internet of Things to launch cyber attacks. The Hajime worm, detected in 2016, also used poorly configured networks to infect smart devices [35]. All the above examples of attacks were carried out because users did not change the default passwords in smart devices. They choose such weak passwords or leave the default credentials as they are just easier to remember [36].

After compromising smart devices, the adversary can set himself up as MITM, relaying and changing the messages exchanged between the various components of the smart environment [37]. In addition, the adversary may launch a phishing attack to steal user credentials. An example of an attack on Google Chromecast was demonstrated by Dan Petro [1]. The researcher exploited a vulnerability that allowed him to send a de-authentication command to the Chromecast, causing the device to disconnect from the local network and open its WIFI hotspot. Subsequently, the researcher was able to replace the content displayed on the smart TV as the Chromecast was connected to it. The same vulnerability could be used, for example, to extract the details of a user's bank account.

Various companion apps allow users to remotely connect and access smart devices. Often, such applications are over-privileged because of an imprecise definition of privileges [38]. For example, in a scenario where only users present in the office should be allowed access to certain smart devices (*e.g.*, smart printers), due to the lack of the separation of privileges, all users who authenticate to the same smart environment with their companion apps receive the same set of permissions despite their physical absence. Users who are not physically present in the office may cause intentional or unintentional disruptions.

2.1.3 Access Control

Each multi-user platform requires access controls to enforce segregation of duties and limit access to sensitive data. A wide range of smart devices is shared between different users in smart environments, including household members or coworkers [39]. In addition to enabling communication between smart devices, the smart hub, and companion app are responsible for providing access to resources (*i.e.*, devices or data) to legitimate users.

[1] `https://www.raspberrypi.com/news/rickmote-rickrolling-chromecast-users/`

Even nowadays, one of the most common authentication mechanisms is the sharing of an access password. Despite its popularity, this method may be impractical in smart environments as it requires users to enter a password or passcode via a dedicated interface such as a keypad or PIN pad, which smart devices often do not have [40]. Alternatively, tag-based methods have been suggested in which the smart environments could rely on technologies such as QR code, near-field communication (NFC), and radio frequency identification (RFID). Again, to deploy these, specialized hardware is required, which is not typically present in constrained smart devices [41].

Since smart environments are dynamic and constantly evolving, their access control mechanisms must be scalable and intuitive. Although standard input interfaces are not available, various solutions exist to take advantage of the heterogeneity of smart devices. In particular, embedded sensors can provide additional input about users of smart environments and this contextual information that can be used to identify them. This type of information may include user characteristics or behavior, as well as their location, and access patterns [42]. The use of context awareness in smart environments has been extensively researched either as a main or an additional authentication factor [41, 43–47]. For example, Ashibani *et al.* [43] have proposed to use contextual information that includes the characteristics of the user along with a password during the authentication process. However, their solution does not verify the physical presence of the user in the environment, so the adversary can deceive the system just by being near the building.

Lu *et al.* [45] have proposed an alternative access control scheme for smart environments that recognizes signal patterns when a user enters or exits a room and determines their physical location based on this. Their solution uses the received signal strength (RSS) to locate the user, but the accuracy of this system depends on the location of the access point. For instance, if it is placed between two rooms, the device may incorrectly determine the user's location.

Contextual information can strengthen access controls by providing unique data points, however, without proving the physical co-location of the user and smart devices, this may not be enough. While the observation that devices in the same location can capture similar contextual information is accurate, relying on properties such as RSS, GPS, etc,

does not provide the required precision [48]. Thus, the research has shifted into utilizing other capabilities of smart devices to capture contextual information.

Overall, existing access control mechanisms are not sufficient to tackle challenges concerning dynamic IoT infrastructures. However, access control in smart environments can be facilitated by using heterogeneous capabilities of smart devices. Interacting with these devices is a robust way to prove the physical co-location of a user and a smart environment.

2.1.4 Cross-channel Interactions

Not only do users interact with smart devices, but these devices also communicate with one another. This type of communication can come from user-configured automations or signal propagation. Smart hubs and companion apps provide interfaces that allow users to control and manage these automations. Whereas such interfaces provide useful feedback to the users, they only contain a partial picture of the actual communication model within the ecosystem. This is because physical signals travel through various channels and can be detected by different sensors. For example, if a user enters a room, the event can be recorded by microphones, accelerometers, temperature sensors, and so on. In other words, the same event or physical interaction can be captured by different devices, using various modalities. Furthermore, the operation of one device may directly or indirectly affect the other, causing, in some cases, a chained reaction. An example scenario might be when a user has configured smart bulbs that turn on when motion is detected in the room and added automation that turns on the washing machine one hour before coming from work. When the washing machine starts to run, it generates vibration, moving itself or nearby objects, which is then picked up by the motion sensor. As result, this sensor turns the lights on even though no one is present in the room.

2.1.5 Detection of Cross-channel Interactions

Recent work has been trying to remedy this by addressing various threats such as the collaborative misuse of smart devices due to the interactive nature of the Internet of Things [49–53]. Since smart environments are dynamic infrastructures, it is necessary

to take a step back and consider the whole picture of such an ecosystem in order to secure it. One possible solution is to model and verify events in such environments to prevent devices from entering unsafe states. IoTSan [54] and IoTCheck [55] examine the state of smart environments using modeling techniques, however, these approaches do not consider physical channels. Similarly, ProvThings [56] detects conflicts and enables the examination of past events through formal model verification. On the other hand, Ruledger [57] uses a distributed ledger to collate triggers and only allows action if there is a valid rule to trigger it. This way, Ruledger detects and prevents unwanted events from occurring in the smart environment. All aforementioned approaches focus on keeping the environment safe by monitoring specific rules and automations. However, none of these solutions address threats posed by cross-channel or user interactions in the smart environment.

Depending on the vendor, some smart devices are supplied with companion apps. Such applications are used to control and monitor devices and their states. Moreover, they connect devices to smart hubs, which allows integrating them into a larger smart ecosystem. Several approaches to detecting cross-channel interactions focus on static analysis of companion apps. For example, IoTMon [15] performs such an analysis on the application's source code to extract the automation components and cross-analyze these to predict cross-channel interactions. It extracts information about the physical channels from the description of the device, using natural language processing (NLP) techniques. Another system, HomeGuard [16], uses similar techniques and adds symbolic execution to extract rules, however, it does not consider physical channels. Similarly, iRuler [17] uses NLP to detect dependencies between automations and based on the extracted data performs model checking.

Unfortunately, these approaches, which are based on static analysis, are prone to generate a large number of false positives as they do not consider hardware limitations or the relative positioning of smart devices that affect signal propagation. Moreover, training such systems requires a lot of time and has to be repeated every time a new device is added, moved, or modified; thus, these systems do not scale. The systems also rely on accurate device descriptions since they utilize NLP to extract physical channels.

This, however, cannot be guaranteed. Interestingly, none of these systems examines the impact of user interference in the environment.

2.2 Biometric Systems

Biometric authentication and identification systems that utilize data collected from mobile and smart devices are broadly categorized into single-biometric or multi-biometric approaches [9, 10]. The systems from the first category collect inputs of a specific type (*e.g.*, sounds, images, acceleration readings) and search for unique patterns. On the other hand, multi-biometric systems combine the data extracted from multiple sources to create unique signatures based on different sensor types. They provide more flexibility and lift several limitations posed by single-biometric systems, including dependency on certain types of equipment and environmental conditions. Moreover, they are less prone to mimicry attacks due to the complexity of spoofing multiple modals simultaneously [58].

2.2.1 Single-biometric systems

The vast majority of existing commercial and non-commercial systems used in smart environment contexts [5, 6, 59, 60] primarily rely on voice recognition to authenticate and identify users. Since these systems are often vulnerable to voice spoofing and hijacking attacks [61–64], research efforts shifted toward hardening voice recognition systems by leveraging anti-spoofing mechanisms like proximity detection or second factors [6, 60].

With the recent development of new types of Internet of Things and wearable devices, the possibility of using unconventional biometric traits has emerged. For instance, Chauhan *et al.* observed that microphones can be used to extract breathing acoustics when the user is present in a smart environment [7]. Similarly, the built-in accelerometers in mobile and IoT devices have been used to characterize gait or human body movements to facilitate authentication [8, 65–67]. These approaches were the first steps taken to explore the full potential of smart environments to turn contextual and behavioral data into biometric traits for seamless authentication.

2.2.2 Multi-biometric systems

To improve adaptability and reduce the inaccuracy of single-biometric systems, various multi-biometric systems have been proposed [9–14]. One approach is to combine two biometric traits to fingerprint users [12–14]. For example, Olazabal *et al.* [13] proposed a biometric authentication system for smart environments that combines voice and facial features. These solutions, however, still require users to actively participate (*e.g.*, by shaking devices or repeating specific hand wave patterns) in the authentication process and rely on the presence of specific sensors in the smart environment. To address such limitations, the MUBAI system [9] employs multiple smart devices to extract various behavioral and contextual features based on well-known biometric traits such as facial features and voice recognition.

2.2.3 Multi-sensor fusion

Multi-sensor fusion systems focus on identifying specific actions using diverse types of inputs supplied by a variety of sensors. Instead of relying on one modality, signals are extracted from different sources [68]. Most of the research in this area has been developed to aid Human Activity Recognition (HAR). Some HAR systems recognize certain types of body movements by analyzing objects' orientation and acceleration [69–71]. Others precisely identify tasks such as cooking or cleaning the house by applying sensor fusion [72, 73]. Using multiple sensor modalities improves the activity recognition performance and creates a better reference frame [74]. A variety of techniques, including different ensemble methods, can be used to combine heterogeneous modalities [68].

2.2.4 Interaction-based biometric systems

Interaction-based biometric systems have emerged from the observation that physical interactions with various objects, including smartphones, smart watches, or other mobile platforms, can uniquely identify users by fully exploiting the heterogeneous nature of built-in sensors. Such systems profile users by collecting relevant information about their unique movements and gestures when they perform certain tasks, including typing, clapping, walking, and such [75–77]. On-device sensors are employed to measure touch

dynamics or user gestures [78–80]. For example, users can be profiled based on how they pick up their phones or how they hold them [81]. Similar techniques have been used in smart environments; however, most of the existing solutions not only require the user to actively participate in the authentication process but also rely on a specific setup. Moreover, instead of a single device, the entire infrastructure can be used to capture these interactions. SenseTribute [82] is an interaction-based occupant identification system that extracts signals from accelerometers and gyroscopes. It ensembles and clusters user activities with supervised and unsupervised machine learning techniques. Although this system is intended to scale as the number of smart devices increases, it has a number of limitations. It focuses on a restricted range of sensor modalities and a single smart environment, with a small number of users. Additionally, it does not use co-located sensors (i.e., sensors located on nearby objects) for fingerprinting users, which constrains its identification performance and renders it more challenging to deploy in existing smart homes or offices, as it imposes restrictions on device placements.

2.2.5 Mimicry Attacks

The most common attacks against biometric systems are mimicry attacks. In these attacks, the adversary tries to impersonate the victim during the authentication process. Depending on the specific system, these attacks can take different forms [83].

Interesting examples of such attacks are attacks on facial recognition systems. Initially, these attacks consisted of wearing a mask or glasses or applying makeup [84–86]. In some other cases, the adversary could only show photos of the authorized user to gain access [87]. Such mechanisms were also attacked using light or infrared [88, 89]. I highlighted this type of biometric system because attacks against it show how challenging it is to create a mechanism that is immune to all mimicry attacks.

Due to the heterogeneity of smart devices, many authentication systems use behavioral or interaction-based biometrics. Typically, this type of biometrics is hard to imitate, as it depends not only on the user's behavioral patterns but also on their unique anatomy. For instance, when considering an authentication scheme based on brief interactions of

smartwatch wearers with IoT devices [90], an attacker needs to pay closer attention to many factors, including gesture speed, strength, and other relevant factors. Since most of these factors are difficult to control, the success of mimicry attacks on such systems is less.

While developing a new authentication method, the first type of mimicry attack tested on such a mechanism is the zero-effort attack [91]. In this attack, the adversary tries to gain access to the system as a legitimate user without actively trying to impersonate this user. For example, by presenting their face or voice, hoping the system will not distinguish them from the legitimate user. This is the first step in determining whether such a mechanism can be deployed. If the attacker manages to gain access most of the time, the method is insufficient. In turn, this may depend on the selected biometric feature, which may not contain enough uniqueness or the selected Machine Learning (ML) model.

The second category of attacks is active mimicry attacks, in which the adversary is actively trying to imitate the victim, *e.g.*, by showing an image of an authorized user's face if the system is based on facial recognition, or by presenting a voice recording if the mechanism authenticates users through voice recognition. Such attacks can challenge biometric systems, especially those that rely on the recognition of a single feature. It is much more difficult for the adversary to imitate many biometric modals, for example, the voice and movement of the victim, at the same time. Liveness detection [92] can be used as a preventive method against a group of attacks in which the adversary presents the system with a false biometric proof (*e.g.*, a photograph of the legitimate user's face).

2.2.6 Metrics

To evaluate the usefulness of the proposed biometric system discussed in this thesis, and analyze its performance, I used two metrics, namely False Acceptance Rate (FAR) and False Reject Rate (FRR) [93].

FAR specifies a rate at which the system falsely accepts a user who is trying to impersonate a legitimate user. On the other hand, FRR describes the rate at which a benign user is falsely rejected by the system. These two metrics can be combined to show FRR at fixed thresholds of FAR, which can be used to tune the biometric system, depending on the security requirements [94]. Better usability can be provided to the

users by minimizing FRR. However, if the focus is on security, higher FRRs and lower FARs are desired. These metrics are also useful to analyze the biometric performance of the system under zero-effort and active attacks.

Furthermore, I use the Receiver Operating Characteristic (ROC) curve, which plots true positive rates (TPRs), which can be expressed as 1 - FRRs, against FARs, to showcase the performance of the biometric system given different classification decision levels. In other words, it shows how accurately the ML model separates the classes.

2.3 Remote Experimentation

Modern science is based on empirical evidence. Such evidence can be gathered through unbiased, rigorous observation or experimentation. In fields like computer science, experiments play a key role in validating hypotheses, studying new phenomena, and designing new systems and protocols. The fundamental factors of credible experimentation are randomization and control [95]; randomization ensures true representation and inclusivity in terms of participants and experimental environment, while control enables researchers to isolate variables and components being studied [96].

2.3.1 Remote Experimentation Practices

Remotely controlled experiments have traditionally been used in the fields of physics and engineering [97–99], in cases when the equipment cannot be accessed during the experiment [100] or as part of distance learning [101]. However, such remote experiments do not typically involve human participants: once the equipment is set up by the researchers, the experiments are conducted and recorded remotely by the same research team. Under lockdown restrictions, many researchers found themselves in a situation in which they needed to conduct an experiment involving human participants, yet they could not be present at the experiment location [102]. With human participants in the loop, the complexity of setting up and conducting remote experiments becomes a new challenge that calls for a new methodological approach. As noted by Zeng *et al.* [30], in order to gain real insights into the interactions of users with other users or devices

in a smart environment, their behavior should be studied in their private environments where they feel comfortable and natural.

2.4 Research Scope

2.4.1 Research Gaps

As previously mentioned, the current access control mechanisms are insufficient for ensuring security in IoT environments. While contextual information can enhance access controls by offering distinctive data points, this may not be sufficient without verifying the physical co-location of the user and smart devices. The standard localization techniques, such as GPS and RFID, do not offer the required precision level and are therefore deemed unreliable. In this dissertation, I investigate the potential of smart devices to identify users based on their physical interactions with said devices, while also verifying the user's co-location.

While the users' identity may be recognized using existing single or multi-biometric systems, many of these systems require users to actively participate in the authentication process, which limits their practicality. Moreover, they heavily rely on the presence of specific sensors within the smart environment, rendering them unsuitable in real world scenarios. My thesis proposes a novel approach that not only eliminates the need for users to actively engage in the authentication process but also utilizes the existing infrastructure to capture relevant biometric data without enforcing a specific setup.

Obtaining realistic data samples is critical for accurately evaluating any biometric system. As participants may exhibit different behaviors in simulated environments, conducting remote experiments in familiar locations, such as their homes or workplaces, may be necessary. However, the field of cybersecurity is relatively new, and therefore lacks standardized experimentation protocols and guidelines. To address this gap, I propose a remote experimentation framework that enables researchers to conduct experiments in the environments comfortable for the participants, including those with mobility disabilities. I designed this framework to be generic, facilitating its use in any type of biometric study.

Finally, while multiple techniques have been discussed for detecting cross-channel interactions among smart devices, most of these solutions depend on static analysis for deriving the interaction model. However, static analysis may fail to capture all dependencies or it may generate a significant number of false positives. Hybrid approaches, on the other hand, are time-consuming and require users to vacate the smart environment during the training phase. To overcome these limitations, I propose a dynamic analysis method that analyzes causality using low-level sensor data and does not necessitate users to leave the environment.

2.4.2 Adversary Model

Within this thesis, I explore two distinct types of systems that analyze interactions with IoT objects, and as such, I present separate threat models for each system. In evaluating the biometric authentication system, I assume that an adversary has physical access to the environment and devices therein, which could potentially be someone with benign intentions such as a child seeking to access restricted TV channels or ordering their favorite food via a smart fridge. Nonetheless, the adversary's primary goal is to mimic the authorized user's movements convincingly enough to persuade the smart environment that they are a legitimate user.

For the second system, I consider an adversary whose objective is to trigger safety-critical automation by exploiting existing cross-channel interactions within the IoT ecosystem. The adversary cannot physically access the IoT infrastructure or compromise the companion apps or an IoT hub, but they can identify the devices present in the environment and exploit their known vulnerabilities. They may also deceive the user into installing a malicious companion app that could suggest and configure new automations that violate safety protocols. In both cases, I assume that the systems are uncompromised during the initial setup and training stages.

3
Preliminary Study

To discover the full potential of heterogeneous interfaces of smart devices for securing smart environments, I started by examining how these interfaces can provide additional information about such environments and their users. Based on the research gap identified in the previous chapter, my first goal was to examine whether these devices could provide evidence of the physical presence of a user in a smart environment. Later, I investigated whether this could be utilized in pairing and authentication, as well as the detection of cross-channel interactions between such devices.

During my preliminary research, I designed the proof-of-concept pairing protocol for smart devices and discussed its advantages and weaknesses. Based on the outcome of the protocol analysis and other conclusions I have drawn, I later developed a user authentication and identification scheme that is based on similar principles.

3.1 Proving Co-location

The ability to localize users in smart environments has become a desired characteristic of emerging access control systems and location-based services [103, 104]. The emergence of more precise localization technologies (*i.e.*, wireless 802.11 mc RTT, GPS, BLE beacons) resulted in incorporating such methods in access control. This is supported by previous studies in which one of the most desired features of access control mechanisms

in smart environments was cited to be the restriction of access to the infrastructure based on physical presence [105]. In such a scenario, the smart hub tries to verify the co-location of a user and a smart environment. The user's co-location with the smart environments is defined as the presence of that user and the smart devices in the same physical location.

Various location-based access control mechanisms for smart environments have been proposed [45, 106, 107]. However, the adoption of these controls in such environments is still very limited due to inaccuracy of co-location detection or scalability and compatibility issues [30]. During my preliminary research, I explored methods to verify users' physical presence in the smart environment as well as discussed how interactions with smart devices can be used as a root to establishing the secure connection between their mobile devices and the smart hub.

Standard approaches for enforcing access control in smart environments include deploying NFC devices in different locations and integrating them with a smart hub [108, 109]. However, such an approach either requires purchasing and configuring additional devices; thus, adding a deployability overhead, or assuming that every smart device is equipped with a built-in NFC module. Moreover, it fails to address scalability, *i.e.*, and it cannot be used to restrict access to individual devices. To solve the problem with the lack of granularity, the Bluetooth Low Energy (BLE) beacons could be attached to every device [110]. However, it is impractical for large smart infrastructures.

Alternatively, to prove physical co-location, distance bounding mechanisms, which restrict the maximum distance between two parties who are communicating, could be utilized. For instance, Pico [111] is an authentication system that removes the need for traditional passwords by utilizing a combination of hardware and software to offer a more secure and user-friendly authentication system. The system includes a USB stick with a secure key that is employed to authenticate the user to a web platform. To log in to a service, the user simply inserts the Pico into their computer's USB port. Despite being a promising solution, it has various limitations. Since the USB stick is a physical object that users must carry with them, it may be inconvenient for some users, particularly those who frequently switch between devices or work remotely. Additionally, if users lose their device, they must change the authentication settings on all web platforms. Furthermore,

if an adversary gains access to the user's computer while the Pico device is connected, they can use it to authenticate themselves. Lastly, widespread adoption of Pico would necessitate significant infrastructure changes by websites and services to support the new authentication system, which could be a significant obstacle. While distance bounding can prevent basic relay attacks, more sophisticated adversaries may be able to bypass the protocol by utilizing advanced techniques like man-in-the-middle attacks. Moreover, distance bounding techniques frequently require specialized equipment and infrastructure, which can be a significant barrier to adoption.

As for scalable smart device pairing, McCune *et al.* [112] proposed using a camera of a mobile device to capture a barcode generated and displayed on a screen of another electronic device. A piece of information included in the barcode can be used to bootstrap key exchange or secure the configuration of smart home devices. However, this solution requires that devices are equipped with displays or cameras, while many smart devices do not have either. For a comprehensive overview of previous work on general device pairing, I refer the reader to [113]. While there have been numerous other approaches to pairing specific smart devices [114], pairing a mobile device to a smart hub introduces a new set of challenges.

Since Augmented Reality (AR) enables the creation of user interfaces that present instructions in a more accessible way, it can improve the interaction between Internet of Things systems and users. Given the challenge of increasing the usability of smart devices, ARIoT [115] is a framework that identifies smart devices that are close to the user and then provides an AR interface to interact with them. The set of interactions is sent to the AR device along with the object recognition instructions by the smart devices. While the authors highlight the importance of portability and scalability in smart systems, the proposed solution uses Bluetooth beacons (as opposed to concurrent odometry and mapping) to locate the nearby devices, which becomes impractical for environments with many heterogeneous smart devices that do not have these specific hardware capabilities. Furthermore, the authors do not discuss how the initial secure connection between the smartphone and the central hub is achieved, which is precisely the focus of my preliminary study.

3.2 Scope

At the initial stage of my research, I tried to answer the following questions:

1. How can the existing smart infrastructure verify the user's physical presence in the smart environment?

2. How can input and output interfaces of smart devices are used to prove the user's physical presence to the smart hub?

3. What is the maximum and minimum entropy that the user interactions with smart devices can provide?

4. How can the smart hub learn the locations of smart devices within the ecosystem?

Since existing access control mechanisms still struggle to scale with dynamically expanding smart infrastructures, I examined how can one utilize the smart devices that are already connected to the smart hub to verify the user's physical presence in the environment. To answer this question, I had to find a way to supply the hub with information about the locations of smart devices in a smart environment. Thus, I examined whether the Augmented Reality (AR) capabilities of modern mobile phones can achieve that with the help of a user. AR technologies allow devices to construct immersive experiences in which virtual objects are naturally overlayed over one's view of the physical world, enabled by novel display technologies and precise environmental mapping. Motion tracking and environmental mapping are achieved by concurrent odometry and mapping (COM) [116]: tracking the position and movement of distinct feature points in the captured camera image and combining this information with measurements from inertial sensors. As a result, the device is able to create a model of its environment while at the same time using the camera and inertial sensors to determine its location and orientation in the generated model.

3.3 Protocol Overview

The proof-of-concept protocol, `ARKey`, is shown in Figure 3.1. The goal of this protocol is to verify that the user who initiated the establishment of the secure connection indeed has physical access to the smart environment. It builds upon the general idea of establishing a secure communication channel using Short Authenticated Strings [117, 118], which I simplified and adapted to the specific considerations of using mobile AR to instruct the user's interactions with connected devices.

To initiate pairing, the user U_A sends their public key P_{KU} to the smart hub SH. In response, SH sends its public key P_{KG} along with the list containing inputs ($inList$) and outputs ($outList$) of connected devices to U_A. Next, SH generates random nonces K_1 and K_2. SH sends K_1 and commits to a strong-hash of K_1, K_2, $inList$, and $outList$, which results in a smaller probability of finding a collision by Adv. U_A calculates a weak-hash H_{W_1} of K_1, P_{KU}, and P_{KG} to generate a list of actions, which consists of interactions with connected devices that U_A has to perform. Since it would take a significant amount of time to perform actions based on the high entropy strong-hash, we use the weak-hash to determine physical interactions that can be performed by U_A. After SH is received and verified expected inputs, it sends K_2 encrypted with the public key of U_A. Next, U_A verifies the strong-hash commitment received in the beginning. SH commits to a weak-hash H_{W_2} of K_1, P_{KU}, and P_{KG} that represents an output. U_A can now verify if the output from SH matches H_{W_2}. If the output does not match, the protocol aborts. Otherwise, the protocol results in establishing an authenticated channel.

3.4 System and Adversary Model

The system model, shown in Figure 3.2, consists of the user U_A, multiple connected devices D_1-D_3, and the smart hub SH. All elements are located in the same environment (*e.g.*, smart home or office). The main goal of this system is to allow control over devices D_i only to those users who are physically present in the smart environment.

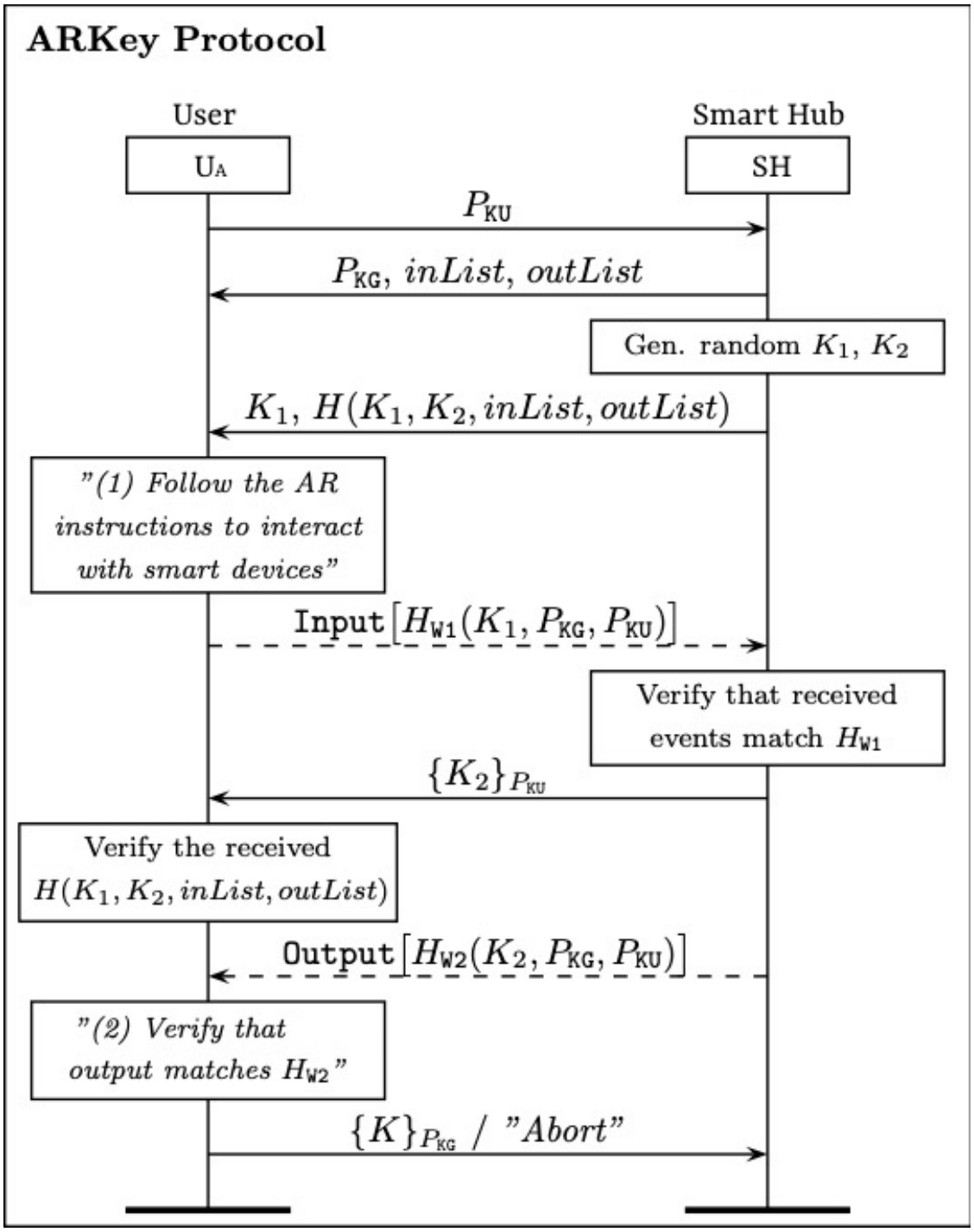


Figure 3.1: Location-based establishment of secure connection. Using the insecure channel, the devices first exchange their public keys and the public information about the connected devices available for input and output. Next, the smart hub SH generates a commitment with nonces K_1 and K_2 and input lists, after which the user performs a set of input actions based on AR guidance. Finally, the gateway opens the second part of the commitment and performs output actions to prove its authenticity by showing the ability to control the connected devices. Finally, user U_A verifies and confirms/aborts the execution in the last message.

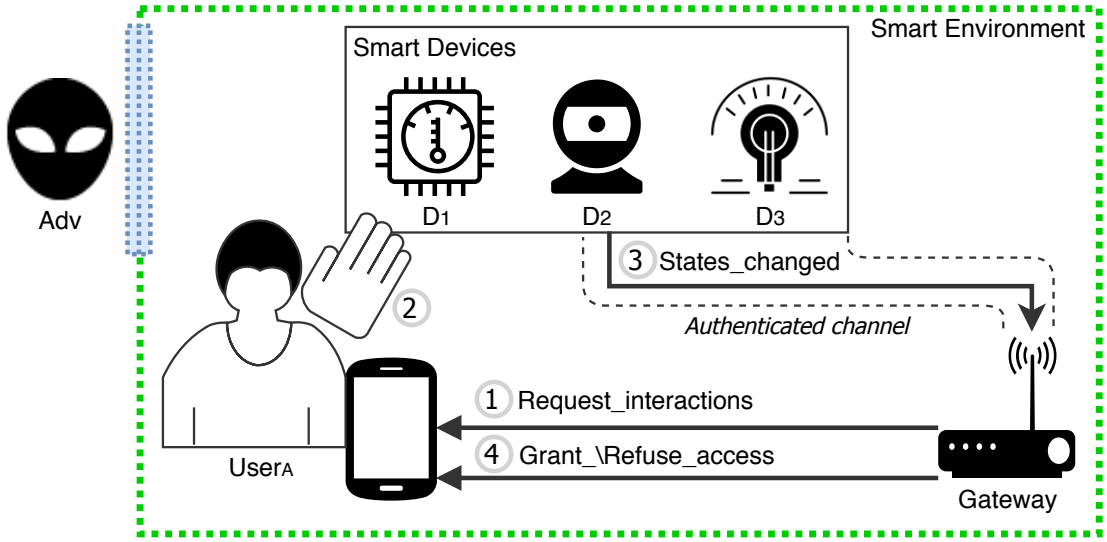


Figure 3.2: Overview of the system model presented in this thesis. After the user U_A requests remote access to smart devices D_1-D_3 from the gateway SH, the gateway asks U_A to prove that he has physical access to D_1-D_3 by instructing the user to interact with these devices (1). U_A uses an out-of-band physical channel (2) to interact with the connected devices. Next, SH uses an authenticated channel to communicate with connected devices to very the occurrence of the requested interactions (3). If the verification has been successful, SH grants remote access to U_A (4). U_A's mobile phone communicates with the gateway via an unauthenticated channel. The adversary is located outside the smart environment; however, he can observe the interior of this environment.

The smart hub serves as an interface that provides access for AR-enabled mobile devices, such as smartphones or AR headsets, to control the connected devices D_i in its vicinity, such as coffee machines, smart printers, thermostats, etc.

The initial wireless connection between the user's mobile device and the smart hub is neither authenticated nor encrypted. The bidirectional connection between the hub and the smart devices is authenticated: the smart hub receives updates when the state of devices changes (*e.g.*, the light switch turns off, thermostat temperature is modified, etc.), and can modify their states. U_A can physically interact with smart devices D_i via an out-of-band physical channel: they can change their state by physically interacting with such devices (*e.g.*, turning light switches on and off, setting the thermostat temperature). U_A's objective is to obtain remote access to smart devices through her mobile device after proving initial co-location with smart devices to the smart hub.

I considered a Dolev-Yao [119] style network adversary who fully controls the initial wireless communication channel between the U_A and SH, but is not physically present

inside the smart environment. However, the adversary can be located in close proximity, know the layout of the environment, and can even visually monitor the environment, *e.g.*, by looking through windows or glass doors. The adversary's goal is to obtain remote control over the devices D_i by falsely proving their physical presence to the hub or to eavesdrop on a legitimate user's communication with the devices (via the smart hub).

I considered denial-of-service attacks and collusion between the legitimate user with the adversary to be out of the scope of this preliminary work. Thus, I focused on the following attack scenarios:

- **Man-in-the-middle attack**: In this scenario, the adversary attempts to obtain a session key by intercepting and modifying the messages exchanged between the legitimate user and SH, resulting in gaining remote access to the smart devices D_1-D_3.

- **Guessing attack**: If the adversary cannot observe a session key in transit, he can launch a brute force guessing attack to try various combinations of the key.

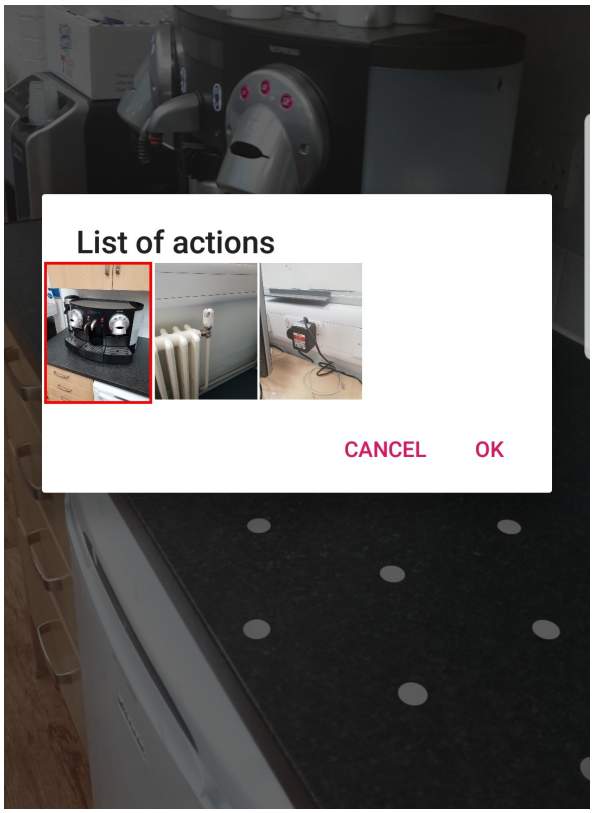
- **Replay attack**: In this scenario, the adversary can replay the messages from the legitimate user to SH to skip the necessary authorization checks.

- **Accidental authorization**: Since the adversary can observe the environment before launching the attack, they can leverage the knowledge of the daily usage of smart devices. This means that the adversary can initiate a key exchange with SH when legitimate users perform their daily operations using smart devices and let SH believes that the signals from the smart devices are the result of input interactions.

3.5 System Prototype

To test the operation of this protocol, I built a prototype of a system that implemented it. This system consists of the following elements:

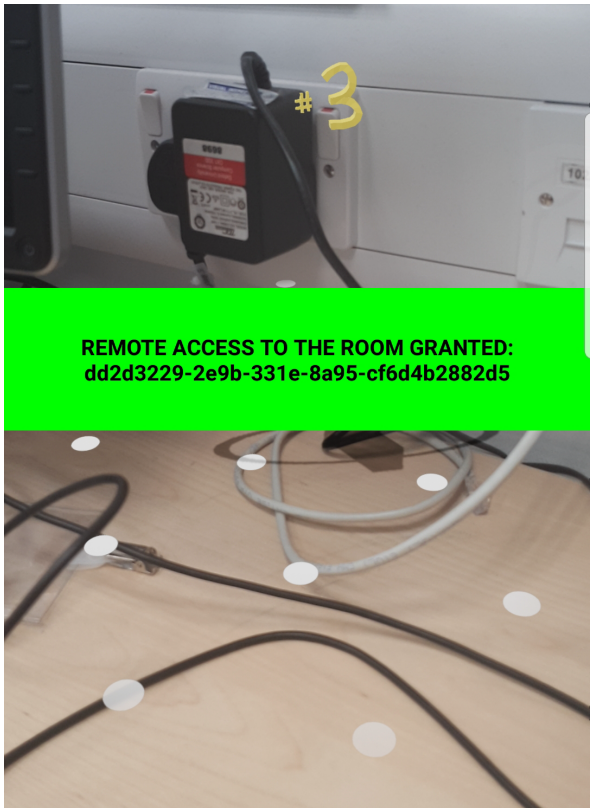
Smart Hub This device manages permissions to access various resources within a smart environment. If U_A wants to remotely control and operate smart devices, a smart



(a) The list of tasks given to the user.



(b) The application shows the first interaction point to the user.



(c) The mobile application informs the user that remote access has been granted.

Figure 3.3: Overview of the user interface. In order to authenticate the exchanged keys, the mobile application indicates the tasks that the user has to perform to prove their physical presence in the smart environment, as well as the actions that the smart hub will perform to authenticate.

hub has to first confirm whether they have physical access to the environment to grant appropriate permissions. Because smart hubs do not know the location of the smart devices within the smart ecosystem, before users can pair their mobile devices with the smart devices, the hub must be supplied with position markers to recognize the physical locations and features of the smart devices in the ecosystem. The enrollment of position markers can be done by the system administrator once during the initial configuration of the smart hub. Smart environments tend to expand rapidly; thus, the system must provide the functionality to enroll the position markers after new devices are added. To avoid accepting position markers sent by an unauthorized user, the smart hub only stores the parameters supplied by the authenticated administrator. The user who wants to obtain remote access to smart devices must perform certain actions requested by the hub. For instance,

the smart hub can ask them to press a specific sequence of buttons on a coffee machine. If they can complete these tasks, this means they have physical access to the machine; therefore, they can be granted remote access.

Mobile App The main function of the mobile application is to provide information on the physical locations of smart devices and their functions to the smart hub, as well as to provide an interface to users. To defend against MITM attacks, the mobile app has to establish and maintain a secure connection with the hub. Such a mobile application creates a virtual representation of the physical world based on recognized feature points. Depending on these points, this representation may vary between various mobile devices. Thus, to give multiple users the possibility of correctly identifying the same locations during the pairing process, the mobile application has to be able to access previously enrolled feature points and attempt to match them with the presented view of the environment. Therefore, these feature points have to be stored on the external server or the smart hub and sent to the mobile app when the user begins the pairing process.

User Experience Although the protocol consists of multiple steps to ensure compliance with security requirements, user interactions are relatively simple. In order to authenticate a new mobile device to the smart hub, the user is prompted to perform a series of tasks (input actions) and verify the set of output actions that the hub performs using the smart devices. The mobile device guides users throughout this process by displaying the list of actions given to the user and indicating the interaction points. As shown in Figure 3.1, the user input actions (the first dotted line) are generated using N_1 and the smart hub output actions (the second dotted line) are computed using N_2.

Figure 3.3 shows a simple scenario where the user is prompted to interact with the smart coffee machine. The instructions (a) and interaction points (b) are indicated by AR pointers on the screen of the user's mobile device. In (b), the user is requested to press the right button on the coffee machine once. After the user performs all the input actions, the hub can confirm the user's physical location in

the environment. In the next step, the user must verify the hub's output actions, *i.e.*, confirm if it changed the states of the smart devices as specified. For instance, the hub may flash an LED bulb or temporarily change the numeric value on the thermostat. When both parties confirm that the specific input and output actions were performed, the user's mobile device displays the outcome of this process, *i.e.*, receiving remote access to the smart devices (c).

To avoid user rushing attacks [120], the `ARKey` system can be extended to read the output from smart devices (*i.e.*, tracking changing colors of the LED bulb) or ask users to enter the number indicated on the display of the smart device (*i.e.*, the number of cups on the coffee machine or temperature on the thermostat).

3.6 Deployment

To assess the feasibility of the prototype solution using current smartphone devices, I implemented and deployed it. I used Raspberry Pi (RPI) boards for the simulation of a smart hub. Moreover, multiple smart devices and appliances were simulated with individual RPIs, and the user interface was implemented as a mobile application developed on Android, using ARCore SDK [121].

The smart hub is running a JavaScript application that creates a nodeJS server. This server implements two main functions: firstly, it plays its role in the protocol, and secondly, enables the previously authenticated user, a system administrator, to determine the location of smart devices, and register them in the system by providing spatial reference points (so the system can locate them), and interaction capabilities.

To simulate the smart devices in the smart ecosystem, I modified the open-source software from Initial State [122] such that the smart hub establishes a secure connection with the mobile device running the `ARKey` application. This also allowed the hub to monitor and modify the state of smart devices, such as a smart fridge, thermostat, or coffee machine that were wirelessly connected to the server running on the hub via their RPI devices.

The mobile app is an Android application written in Java using ARCore 1.2 SDK and the Sceneform API. This app is responsible for providing the AR interface, which guides interactions between users and smart devices. As mentioned earlier, the AR technology helps the administrator to enroll the position markers of the smart devices, and establish a secure connection with the smart hub by following the protocol. Overall, the main function of this application is to allow the user to pair their mobile device with the smart hub to obtain remote access to the smart devices in the environment.

When a new device joins the smart ecosystem, the hub does not know its physical location, which must be provided after the device is set up. When registering a new device, the administrator points the camera of his mobile phone to that device, and the application generates position markers, which are spatial reference points, for this device, using the smartphone's AR capabilities. These reference points are eventually sent to and stored on the smart hub.

During the pairing process, as shown in Figure 3.3, the mobile application identifies previously registered position markers. It then displays a list of actions that users must perform. Above the recognized point of interaction, the user sees a short instruction or number (*e.g.* press the coffee machine button once). After executing the instructions, the user performs tasks that the smart hub can detect by checking the status of each smart device. As a result, a secure connection to the smart hub is successfully established and the user gains remote access to operate the smart devices in the smart environment. The mobile application informs the user about the successful execution of the protocol.

3.7 Evaluation

In this section, I evaluate the implementation of the operation of the proof-of-concept protocol as well as the technological readiness of the ARCore [121] SDK.

3.7.1 Security Analysis

In this section, we analyze the security guarantees provided by `ARKey`. This system has been designed to protect users from the four attacks discussed in Section 3.4. As mentioned, `ARKey` must prevent the adversary from obtaining a session key K exchanged

between SH and legitimate users after the authenticated channel has been established. Moreover, the system must protect against replay and accidental authorization attacks that would allow the adversary to obtain remote access using the knowledge of the daily usage of smart devices in the environment.

Initially, SH and U_A exchange messages over an unauthenticated channel. To protect the integrity of the messages and avoid impersonation attacks, `ARKey` generates a strong commitment using a sufficiently secure hash function. The use of such a function makes it infeasible for the adversary to find collisions. The adversary can calculate H_{W_1}, however, they cannot use an interface of a smart device to input it since they have no access to the smart environment. A low entropy weak-hashes H_{W_1} and H_{W_2} bind the identities of SH and U_A to the generation of the input/output actions so that the adversary cannot impersonate them. `ARKey`'s protocol enables bidirectional verification between SH and U_A by requesting input/output actions from both parties.

Furthermore, the adversary may attempt to guess the session key K. However, after `ARKey` establishes an authenticated channel between SH and U_A, it generates an AES-256 bit symmetric key used for further communication between them. The adversary may also try to resend messages exchanged between SH and U_A. To prevent that, `ARKey`'s protocol uses freshly generated nonces K_1 and K_2 to determine the list of input/output actions that U_A and respectively SH have to execute, making it infeasible for the adversary to use previously exchanged messages.

Lastly, since we consider scenarios where the fluctuation of individuals is high, certain types of interactions between smart devices and users may occur frequently. If those interactions are included in the list of actions sent by SH, another user can unintentionally perform them on behalf of the person who initially requested remote access. To prevent the adversary from exploiting this opportunity, SH will discard and regenerate nonces K_1 and K_2 if the actions on the generated list happen regularly in the smart environment. To achieve it, the SH stores and updates the list containing the usage statistics. This is discussed in more detail in Section 3.7.3.

Table 3.1: The table shows examples of smart devices that can be used as either input or output actions. It also presents the number of possible smart device interactions that add up to achieve the overall entropy.

Devices	Device interactions	# States	Input	Output
Thermostat	5 to 30°C (in 1°C increments)	25	✓	✓
Window shade	Up/half-up/middle/half-down/down	4	✓	✓
Coffee machine	1-12 cups (in 1 cup increments)	11	✓	✓
Light	On/off	1	✓	✓
	Blue/green/red/yellow	+4	✗	✓
Door	Open/closed	1	✓	✗
Window	Open/closed	1	✓	✗
Doorbell	Doorbell/motion alarm	2	✗	✓
Smart button	Click/double click/long press	3	✓	✗
Total states			46	47

3.7.2 Entropy of User Interactions

Given the lack of input and output capabilities of existing smart hubs, which could be used for device pairing or authentication, the proposed protocol uses existing smart devices as input and output channels. Based on the smart office configuration in [49], I enumerate some of the devices that could be used for such communication, along with the number of states that they can be easily set to by the user (input) and states that can easily be read by the user (output).

As Table 3.1 shows, these devices sum up to 46 different states for input and 47 states for output. Since for each input or output action the user needs to set, change, or observe the state of one of the devices, this yields a total of about 6 bits of entropy that are communicated with each action in this simple setup. I note here that I chose only relatively simple devices, *e.g.*, foregoing using any displays for output or voice for input.

Depending on the expected level of security, the system can mandate various input actions from the user, or perform several output actions and require them to be verified. As an example, if the required list of actions consists of more than one input/output, then the resulting entropy is the sum of individual actions, yielding *e.g.*, for 3 actions (press

Table 3.2: The table shows commonly recurring sequences of actions in the smart office dataset from [49]. I determined that there are seven common sequences repeatedly occurring within five-minute intervals over the two weeks covered by this dataset.

Sequence	# Occurrences
Door/Fridge/Coffee	8
Doorbell/Door/Fridge	6
Doorbell/Door/Light	17
Door/Light/Window	16
Door/Light	84
Doorbell/Door	49
Window/Window shade	10

button X, set the thermostat to 20 degrees, turn on the light) more than 15 bits of entropy. Given that the adversary can only succeed by actively participating in an existing pairing attempt, he has no better way of succeeding than 1 over the total entropy of combinations, which can be sufficiently low for most practical purposes.

3.7.3 Recurring sequences

I also note that certain actions are repeatable, which can result in incorrectly confirming the user's co-location with the smart environment. A smart hub may request sequences of tasks that occur often and thus grant access when someone else is carrying out tasks unknowingly. To this end, I analyze the Peeves [49] smart office dataset. This data was gathered over the course of two weeks in a smart office setup consisting of twelve different smart devices. Table 3.2 shows that there are seven such repetitive sequences in this dataset. The smart hub may decide to discard repetitive sequences based on their occurrences. `ARKey` can be configured to eliminate such commonly recurring sequences based on the analysis of the device's usage data.

The system selects the number of tasks based on the entropy these actions can provide. For instance, in the discussed dataset, the most common sequence is Door/Light with 84 occurrences over two weeks. However, this sequence gives only $\sim$5 bits of entropy; thus, it would not be given to a user. Moreover, in smart environments with more devices, the probability of giving the user a certain sequence decreases.

Table 3.3: The table shows the average number of milliseconds to generate and load spatial reference points using the ARCore Cloud Anchor service. The average and standard deviation values are calculated from generating and loading 12 different spatial markers. Six of the markers were located close to the previous marker, while the other six anchors required that the device scans a never-seen part of the room.

	Enrolling (ms)	Matching (ms)
Average	10581	3396
Standard Deviation	2905	4290

3.7.4 AR Performance

In this section, I discuss two aspects of the performance of the AR framework. First, I focus on how long it takes to load the position markers. Then I evaluate the spatial precision.

Loading Time To enroll a position marker, `ARKey` sends the virtual object's pose as well as information about the user's physical surroundings to the ARCore Cloud Anchor service. During pairing, `ARKey` connects to the same service to match the previously provided environmental data with current ones. I measured the time taken to perform enrolling and matching operations. Table 3.3 shows that it takes 10581 ms on average to enroll a position marker. The standard deviation is 2905 ms. The time taken to enroll the position marker is affected by the surface on which the user is trying to place it. The more distinctive points the application can detect, the faster it can register the position marker.

`ARKey` takes 3396 ms on average to project a position marker. The standard deviation is 4290 ms. This standard deviation is affected by the time spent on finding a matching plane. I enrolled six position markers that were located on the same plane or very close to each other. It took on average 374 ms to project these markers. Therefore, I observed that this operation is significantly faster than enrolling the position markers.

Both operations are affected by the same environmental factors including lighting and textures of different surfaces.

Spatial Precision I also assess the extent to which the current state-of-the-art mobile AR technology allows for deployment in the context of guidance for the user's interaction with smart devices. I measure the precision with which `ARKey` can recreate a previously placed virtual object at the same location by rendering the object 10 times in different runs. The coordinates of the object were located by `ARKey` on average to a distance of 4.55 centimeters from its original physical location. The standard deviation was 1.9805 centimeters.

3.8 Summary

Due to the rapidly growing adoption of smart devices that assist users in everyday activities, novel scalable and intuitive mechanisms to facilitate management and access control are needed. During the preliminary study, I prototyped `ARKey`, the proof-of-concept protocol that explores the possibility of using a user's physical presence in the smart environment to facilitate location-based access control. This system uses an Augmented Reality (AR) interface to guide users through interactions with smart devices, which serve as input and output channels to communicate with the smart hub.

A large number of smart devices have constrained interfaces, so well-known mechanisms, such as passwords, are impossible to deploy [123]. `ARKey` has no need for additional equipment, it can scale easily, and a list of interactions generated for a user is based on ephemeral keys. Additionally, this solution leverages the dynamic nature of an IoT infrastructure to increase the level of security by providing larger entropy of interactions. It is also a suitable solution for constrained smart devices that are manufactured to optimize power consumption and costs [107]. By building a functional prototype in a deployed smart environment, I show that different combinations of physical interactions can provide sufficient entropy to resist an adversary who tries to obtain unauthorized remote access to the smart environment. However, due to certain limitations of AR technology and the overall impracticality of this system, I consider it not deployable in modern smart environments.

3.9 Lessons Learned

In this section, I discuss various disadvantages of the `ARKey` protocol and its implementation. In addition, I describe how this preliminary study shaped my future research projects. The first limitation is related to AR technology. ARCore cannot correctly recognize reference points when the mobile device is rotated too rapidly, which may happen when the user walks or performs requested actions. This occurs due to visual-inertial tracking limitations of this framework, *i.e.*, the camera image becomes too blurry to recognize feature points. Furthermore, it cannot easily detect surfaces without distinctive textures since its detection mechanism is based on recognizing distinctive points. For instance, ARCore cannot detect spatial reference points on a white wall or glass.

Apart from the physical limitations of the AR framework, `ARKey` lacks the optimal method of revoking remote access privileges. In order not to limit the usability of this protocol, such a system cannot ask users to prove their physical co-location every time they want to obtain remote access. Therefore, the mechanism of revocation would have to depend on time. For example, the system would have to take into account the time when the user was not observed in a smart environment or find a time window in which, if the rights to control remote access are not used, the permissions would be withdrawn. Finding this optimal time window can, however, be difficult. If the user goes on vacation and still wants to remotely access some smart devices, including smart cameras or thermostats, the system should accommodate this. However, if we consider the smart office scenario in which we want to provide the right to remote access to temporary employees and cancel them immediately, a larger time window would not be optimal.

In addition, even if the system asks users to perform some known interactions with smart devices, I still do not consider it a seamless access control mechanism. This is because users must actively engage in the pairing process, which again impacts usability. Hence, my new objective became to design an access control method that will not require the users to perform any actions beyond their regular activities.

Overall, by designing and building this prototype solution, I realized that this pairing method may not be able to take full advantage of the heterogeneity of smart environments,

but a user authentication scheme based on interactions with smart devices may. During this preliminary study, I looked at how users interact with smart devices. By analyzing the recurring sequences of interactions, I noticed that specific users tend to perform certain actions more frequently. Moreover, by looking at the data traces, although they performed similar actions on smart devices (*e.g.*, open the smart fridge), the sensor readings varied. This has inspired me to examine how unique user interactions with smart devices can be. Furthermore, realizing how many different input and output states each device has; hence, how many different interactions one can perform with an individual smart device, prompted me to wonder how many diverse data points we can obtain from a single interaction.

Due to my experience and knowledge of AR technology, I helped design and evaluate a continuous authentication scheme for AR-head mounted displays [20]. This work shows that users can be profiled by their unique way of interacting with 3D objects. These interactions are recorded by diverse inertial sensors of AR headsets. The proposed system reached an accuracy of almost 93%, which proves that an interaction-based authentication scheme is a viable solution. Based on this work and my preliminary research, I later focused on developing secure and scalable interaction-based identification and authentication method.

Finally, by comparing different devices and their input and output capabilities, I realized that although these devices serve different purposes, they share multiple channels through which they can communicate and influence each other. The lack of awareness of this mutual impact can lead to certain violations in smart environments. Thus, I decided to examine how I can tackle this challenge by modeling cross-channel interactions between smart devices to give users appropriate feedback.

4

Plug-and-play: Framework for Remote Experimentation

Contents

4.1 Motivation 47
4.2 Related Work 48
4.2.1 Controlled Field Studies 49
4.3 Overview of the Remote Experimentation Framework 50
4.3.1 Goals and Requirements 50
4.3.2 Framework Architecture 52
4.4 Case Study 59
4.4.1 Architecture 61
4.4.2 Survey Results 65
4.5 Discussion 67
4.5.1 Research Ethics 67
4.5.2 Lessons Learned 70

4.1 Motivation

To enable the study of access control systems based on user interactions with smart devices, it was necessary to select a data collection strategy that would allow obtaining samples that realistically reflect the natural behavior of users in such environments. The outbreak of the COVID-19 pandemic had a significant impact on many research activities, including in-lab experiments and user studies [124]. Many governments

enforced national lockdowns, which prevented direct interaction between researchers and experiment participants. In the field of cyber security, one of the most affected areas was biometric research due to the need to collect samples from human participants.

As the lockdowns highlighted weaknesses in many areas of human activity, they also showed that some well-established methodologies and experimental frameworks are insufficient for comprehensive data collection and analysis. When studying smart homes or smart office ecosystems, lab experimentation yields data that are less realistic than the data that would be obtained from sensors *in situ*, with their background noises and genuine agent behavior. The processes used to recruit human participants can also restrict the realism of the study. In practice, many studies recruit a homogeneous cohort of university staff and students, many of whom will be inexperienced in the use of the equipment or the ecosystem under test. This also limits the users in the study to a particular place of residence, region, and likely to a certain age group and occupation; other user groups are excluded completely, such as those with disabilities or prohibitively busy schedules.

A possible solution to the problems above is *remote experimentation*. Such experiments might be conducted at scale and enrich the cyber security community with larger, more comprehensive datasets that are currently lacking [125].

However, to set up a new remote experimentation framework, several technical, organizational, and social factors must be taken into account. It is crucial to design a system that is both efficient and scalable while ensuring that it is reliable enough to guarantee the reproducibility of results. Data security and privacy are also a significant concern when dealing with remote experimentation since data must be transferred between different locations. Therefore, ensuring the confidentiality, integrity, and availability of data must be a primary focus during the design phase. Additionally, standardization and interoperability of tools and platforms are necessary to facilitate remote experimentation across different institutions and research domains.

4.2 Related Work

Unlike other research fields, cyber security has very few formalized frameworks or standards for conducting experiments with human participants [126]. The challenges in

developing such uniform standards include (a) ensuring the privacy of the participants as they very often share their private information, and their physical safety, (b) accounting for heterogeneous equipment needed to simulate various ecosystems (*e.g.*, smart homes) as well as the maintenance of such infrastructures [127], and (c) creating a realistic setting for an experiment since security controls should be tested in an environment that is similar to the real one.

4.2.1 Controlled Field Studies

Constructing a realistic environment during the experimentation phase allows for the examination of participants' natural behavior, which is crucial for proper validation of the performance of proposed solutions and models [128]. Classical, supervised environments may provide controlled, repeatable data collection; however, as opposed to *field experiments*, they can neither reach user groups that have no access to the lab due to geographic, medical, or other reasons, nor account for a full range of factors affecting the system under test [129]. Field experiments not only better capture the natural behavior of the users and their potential interactions with the system, but allow for a broader set of experimental settings that are not constrained by a given lab [130]. This makes field experiments the default choice when the research focus is on the interaction between users and systems, products, or equipment [131].

I consider an environment to be *remote* if the researcher conducting the experiment has no physical access to it during the experiment. In some cases, a remote environment can be accessed by the researcher before or after the experiment (*e.g.*, a lab or an office space); in other cases, it cannot be accessed by the researcher at any time (*e.g.*, a participant's private house). A *remote experiment* is an experiment that (a) is conducted in a remote environment, (b) involves human participants, and (c) uses equipment that records certain actions of the participants (movements, interactions with the equipment, *etc.*).

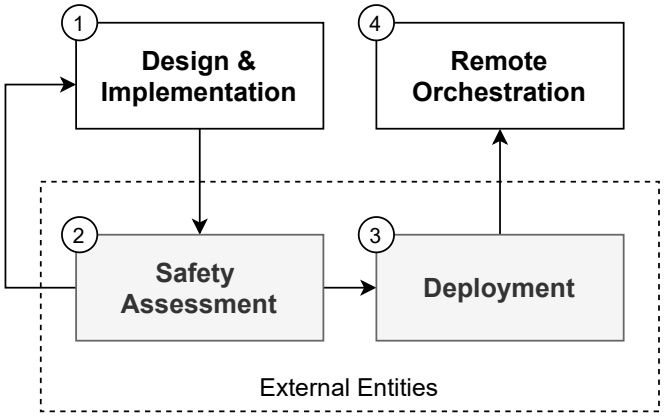


Figure 4.1: The four phases of the proposed framework. Phases 2 and 3 are carried out by external entities such as the institutional review board and experiment participants.

4.3 Overview of the Remote Experimentation Framework

This section describes the proposed framework for remote experimentation and the goals and requirements that drove its development. The framework is divided into four phases, and each phase is broken down into activities that define the experimentation process.

Figure 4.1 shows an overview of the framework. In Phase 1, the researchers refine the experimental design and then implement the tools necessary to conduct the experiment remotely. In the next phase, the risk and safety assessment is carried out by the institutional review board. Feedback from the review process should be incorporated into the design and implementation phases; thus, Phases 1 and 2 cycle until both are satisfied. Once all the requirements are met, the equipment can be transported to a remote site. In Phase 4, the researchers remotely supervise and collect data.

4.3.1 Goals and Requirements

This framework aims to systematize remote experimentation to achieve the following three goals:

G1. **Systematic Design and Deployment.** Remote experiments are conducted outside of a laboratory, in locations such as people's homes, workplaces, or anywhere they spend their free time. These are uncontrolled environments, so tests must be structured with respect to which variables can be controlled and which cannot.

G2. **Realistic, Reproducible Data.** The possibility of conducting experiments in remote environments is important as it allows to study the natural behavior and reactions of users. Thus, data collected during such experiments reflect reality better than data collected in an artificial setting such as a laboratory. Furthermore, the degree of control introduced by the remote experimentation framework enables reproducible, repeatable data collection [129].

G3. **No In-person Interaction.** Not undertaking the experiment in a laboratory allows for the recruitment of participants previously excluded from user studies due to their inability to travel, whether due to disability, proximity, or scheduling. While this has improved over the last decade for people with various disabilities in the field of human-computer interaction [132], it is still a fairly neglected topic in other fields of computer science. Furthermore, such an approach can help in more specific scenarios where external conditions, such as a pandemic, limit the possibility of direct interaction with users.

Moreover, it has the following requirements. Each relates to the entire experimentation process, not to a specific phase, and can be adapted to different use cases.

R1. **Safety.** Ensuring the safety of participants is well established in research facilities, but this is not the case for remote environments. While the researchers cannot influence the general safety conditions of people's private homes, they can identify potential risks and mitigate them wherever possible. The instructions that they supply to participants should also cover safety procedures regarding the use of electronic components and activities pertaining to the experiment.

R2. **Ease of Use.** In contrast to standard experimentation, participants in a remote experiment may have to engage actively in setting up and configuring the testbed. It is important to consider ease of use, both in terms of initial configuration and performing the tasks of the experiment. The equipment given to participants must be easily configurable and durable. The instructions for both setting up and using the equipment must be clear and unambiguous to ensure that the experiment runs smoothly and yields concrete results.

R3. **Ease of Maintenance.** Without a researcher being present, preparations need to be made for troubleshooting and maintenance. Devices and tools should be set with sensible default configurations such that they start in (and reboot to) a usable state. All equipment should be checked before being shipped, online resources should be available, and participants should have ways to contact the researcher for additional support. Such mechanisms must be comfortable for participants to use.

R4. **Remote Supervision and Data Collection.** The researchers must be able to monitor the progress of the experiment. In some cases, this will be in real-time to provide immediate feedback to the participants; in others, the researchers may just wish to ensure that certain milestones were met during the experiment.

R5. **Transmission and Data Security.** There is always a risk that equipment in an uncontrolled environment will get damaged, so all collected data must be transmitted to and stored on a separate device. Backing up the data to an online server is also important because data loss endangers the integrity of the experiment. To ensure the security of the data, all communication between nodes, including backup servers, electronic devices, and other platforms, should be secured by utilizing standard data transmission protocols and tools. Moreover, access to the data on the backup server and other storage units should be protected by appropriate protection mechanisms such as strong passwords and two-factor authentication. Maintaining privacy is of utmost importance in remote experiments, as they often entail the transfer of sensitive data between various locations. Such data may include personal information about the participants or researchers, which makes it susceptible to interception, leakage, or other types of security breaches during transmission. As a result, it is critical to safeguard the confidentiality, integrity, and availability of data to prevent unauthorized access or manipulation.

4.3.2 Framework Architecture

In this section, I provide a detailed description of the four main phases of the remote experimentation framework.

Phase 1: Design & Implementation

This phase consists of two main activities: the design activity, which then guides the implementation activity.

Design. This phase begins with defining the goals and scope of the experiment. It includes describing the desired results as well as identifying the parameters and limitations of the experiment. These tasks are encapsulated in the design activity to ensure an understanding of all key factors that have to be addressed during the implementation activity. More specifically, this incorporates determining what is to be measured, what is to be controlled, how these things will be done, and what is the expected output. Answering these questions will help to obtain information not only about the data and its format but also about participants and environmental factors that need to be involved in the experiment. Thus, the design activity is concluded by preparing extensive documentation that includes the description of experimental and control groups, test sites, test cases, and physical configuration outlines.

The structure of remote experiments allows for the inclusion of people who would otherwise be marginalized. This includes people who cannot travel to research facilities due to distance or mobility issues. Depending on the assumptions made about the experimental and control groups, standard recruiting approaches such as via newsletters or word-of-mouth may not be sufficient as they limit the reach. Alternatives such as recruiting through social media or gatekeepers (*i.e.*, people or institutions with access to the target population) should be considered [133].

Choosing the right location for a remote experiment is another key factor. For example, in many types of research, such as biometrics or activity recognition, it is important to represent the participant's behavior realistically. The research laboratory is an artificial environment that influences the behavior of the participant, often causing the person not to behave naturally. When conducting a remote experiment, a natural environment in which the user feels comfortable can be selected. Examples of such environments could be users' homes, workplaces, or places where they spend their free time. In these environments, however, there is a privacy aspect to consider. Participants should be able to decide when data is collected. Controllers, which are the components that allow

participants to control the entire process, should be introduced. On the other hand, researchers should be able to monitor the status of the experiment and the equipment. These two aspects need to be carefully balanced to avoid interference in the experiment and invasion of participant privacy.

The next step is to prepare test scenarios that will allow for the collection of the necessary data from a target group. These scenarios should be straightforward to avoid introducing too much complexity, which could result in the participant's inability to complete the experiment. Because such experiments are conducted in uncontrolled environments, all uncontrolled parameters should be considered because they may affect the outcome of the experiment. In addition to considering strategies to avoid noise in the data, strategies for handling incomplete data should also be incorporated into this phase. After designing the test cases, researchers should prepare a user manual that guides participants through the phases of deployment and experimentation. For this task, consider what will be best for the participants to accommodate their needs, such as video and audio tutorials and jargon-free language.

Once all parameters are defined and potential constraints identified, an outline of the physical configuration should be prepared to help guide the construction of software and hardware components required to deploy the experiment at a test site. Such an outline can be created in the form of a list of necessary physical components and a description of their communication channels. It could also be a diagram or a combination of both. Designing such an outline helps to understand which elements should be given special attention. Researchers must ensure that the data collected from the participants is transmitted and stored in a secure manner. Many secure communication protocols and tools exist to achieve this goal. In addition, it should be assumed that if the testbed requires internet access, some environments will not have network access points (and some participants will not want to connect to their private network), so alternatives should be considered, such as providing an access point.

Implementation. The second activity takes place after all the design aspects are defined. During the implementation activity, the tools to collect data are either created or selected from the pool of existing instruments. First, researchers have to consider which platforms

the participants are familiar with. For instance, when building the controller, researchers might consider using a mobile or desktop application that will allow the participants to easily manage data collection. To protect their data, researchers should use secure communication protocols such as virtual private networks and enable data encryption on the nodes that are responsible for storing the data. Moreover, if the testbed consists of distributed components, aspects like temporal synchronization and traffic balancing have to be addressed. Additionally, the tools should allow both participants and researchers to pinpoint who is currently participating in the experiment to ensure correct data labeling.

Another important aspect is the delivery of the equipment to the test site. Upon delivery, some components may break, so spare components should be included. It is also important to focus on the usability of such tools as it will help to create a suitable environment to conduct remote experiments safely. For example, the equipment should be as self-contained as possible with all software pre-installed, so as to minimize the possibility of errors in setting up.

The last important aspect in this phase is creating a backup. Tools can be programmed to transfer data to a remote server or store them locally. In each of these cases, protecting the data from corruption or transmission issues is crucial. Overall, good design and implementation can help reduce the likelihood of noise being introduced into the data due to external, uncontrolled factors.

Phase 2: Safety Assessment

In this phase, risk and safety assessments are conducted by the responsible institutional review board that reviews the methods proposed for carrying out the experiments. This step should never be neglected as practices deemed to be unethical can result in serious consequences.[1] Each research institution has different procedures to grant formal approval. The purpose is to identify the risks in every aspect of the experiment and to ensure that they are addressed.

Participants. It is important that all participants are able to understand fully and give informed consent to their participation in the experiment. If the study involves

[1] A paper was withdrawn from IEEE S&P 2021 for such a violation, https://www.ieee-security.org/TC/SP2021/downloads/2021_PC_Statement.pdf

any vulnerable people (*e.g.*, based on age, health, or mental impairment), this should be declared and the risks enumerated. In remote experiments, one must also be cognisant that others, including children and pets, may have access to the remote environment, and assess how they might be impacted by and how they might interact with the equipment (*e.g.*, magnets may need to be shielded so as not to adversely affect medical implants). Depending on the type of experiment, this may necessitate strict eligibility criteria to reject unsuitable environments altogether.

Participants must feel safe and comfortable. Equipment should not pose an unnecessary burden to their everyday life and there should be a mechanism by which they can revoke consent if they so choose. Consideration should be given to how the equipment is shipped to and from participants and how it may be collected early if the participant withdraws from the study.

Testbed. The equipment provided to the participants must be safe to set up and use and must be tested and cleaned in between each deployment to ensure its continued operational safety. Electrical devices should be tested professionally to protect against catastrophic failure (typically, this service can be arranged via an IT department). Does any component pose a choking risk to small children? Are there any sharp edges, open circuits, or loose wires that might cause injury or be pulled by a pet? If such things are necessary for the experiment, and cannot be encased safely, a strategy must be in place to mitigate the risks they pose (such as eligibility criteria, placing objects at height, safety labeling, *etc.*).

Data. The participants should know what data are being collected and how. To protect the confidentiality, data should be stored and transmitted encrypted. If stored unencrypted on capture devices, the data should be wiped before the devices are shipped back to the researchers (unless the researchers collect them directly) for protection in transit. To protect the integrity, data should be backed up automatically, preferably on an external device or server. The data should be accessible only by authorized users, revocable if necessary, and anonymized before publication.

Care must be taken not to capture more data than specified, such as from other people using the remote environment. This may require that devices be easily turned on and off and that they display their operational state at all times.

Phase 3: Deployment

The main difference from standard experimentation approaches lies in Phase 3. Typically, such experiments are deployed and conducted in a dedicated location within a research facility. This gives researchers complete control over the physical configuration of their hardware or software and allows them to detect and quickly fix any errors or technical issues. Additionally, they can monitor the course of the experiment, interrupt it or repeat it when necessary. However, in the case of remote experiments, all these tasks are carried out by the participants of the experiment. Even if the researchers can guide them to some extent through a video or audio conversation, they will be unable to physically solve the technical issues.

Supporting Instructions. For this reason, the preparation of appropriate instructional materials in the design and implementation phase significantly influences the course of the deployment phase. If the participants cannot follow the instructions and operate the equipment, the experiment cannot continue. The basic assumption should be that the participants do not have any experience in operating the equipment, so the testbed setup should be simplified. Moreover, the deployment should be done within a time frame specified in the information sheet given to participants at the beginning of the experiment. Depending on the design of the testbed, its configuration can consume a significant amount of time, and researchers must consider that the deployment takes place in external environments, which may cause problems with the regular use of such spaces. This may cause some participants to withdraw from the experiment, which may impact the outcome and the cost of running such an experiment.

Physical Setup. Another important aspect is the unified physical configuration of the testbed. Depending on the type of experiment, a consistent physical arrangement of the equipment across users may affect the outcome. The researchers may want each participant to use the same device configuration and similar locations and orientations to compare readings. The instructions should be sufficiently strict to guide participants during the deployment phase.

After completing the initial testbed deployment, the participants should verify whether the infrastructure is fully operational. One approach to do so could be to equip the

controller with an interface that informs participants directly about the status of the software and hardware components of the testbed. Another approach would be for researchers to provide feedback to participants after verifying the state of the test environment using remote supervision tools.

Testing. Before starting the experiment, participants should conduct several test trials to become familiar with the equipment and tasks. This will also allow researchers to verify that their supervision tools are working and the system can capture and back up the data. At the end of this phase, each participant involved in testbed deployment should be asked about their experience. This will not only allow for improvements of certain components but also has the potential to reveal problems that would otherwise be difficult to identify based on the data collected alone. Moreover, the researchers could ask the participants to provide supplementary materials, such as a brief description of the location of individual testbed components or photos. The manner of sharing such materials should continue to preserve privacy.

Phase 4: Remote Orchestration

The final phase covers two main activities, including remote supervision of the experiment and data collection.

Remote Supervision. The first activity concerns the ability to monitor the state of the testbed at the test site remotely without violating the privacy of the participants. This includes verifying that all testbed components are properly configured before starting the experiment. This verification should be performed at the component level in order to pinpoint the exact source of a possible failure. In addition, researchers should be able to immediately detect problems not only with the infrastructure but also with the course of the experiment. When the experiment is performed incorrectly, the researchers should be able to interrupt it and give the participants feedback. A dedicated channel should be established between the researchers and participants so that they can communicate their feedback (and any problems) remotely; this could be integrated into the equipment, such as with a controller interface, or it could use third-party software such as voice or video calls or a secure messenger app. They should also be able to resume the experiment if

any issue occurs. Enabling the participants to stop and restart the experiment provides time flexibility and a means to handle unexpected interruptions. These are not typical problems when experiments are conducted in a research facility; remote experiments, on the other hand, can be deployed in spaces that can be used by a large number of people, so such flexibility is essential.

Data Collection. The second activity in this phase concerns real-time data collection. Data from an experiment can be stored in two ways, locally on the testbed components provided to participants or remotely on a server. I recommend that one of these methods is used as the primary storage point for data and the other as a backup. This is important as data transmission can be unreliable in environments without good network coverage. It should also be assumed that there are environments that do not have internet access points, so researchers should provide the appropriate equipment if the testbed transmits the data to a remote server.

Depending on the nature of the research project, the experiment data can be retained for a certain period of time. As stated in Phase 2, the participants of the experiment should be informed in what form and for how long their data will be stored. They should be provided with information about the anonymization of their data, any third parties that may use their data, and the possibility of withdrawing from the experiment, which will result in the deletion of their data. The participants should find out when they can withdraw their participation and for how long their data will be kept. Consequently, data management must be part of the data collection activities to ensure that they are stored and processed in line with the information provided to participants. This means not only that the data must be secured during transmission, but also when stored. Overall, this phase helps to avoid incomplete or incorrect sampling as well as procedural errors due to incorrect transmission and storage of data.

4.4 Case Study

In 2020, social interactions and public gatherings were significantly limited. Due to the restrictions, it was not possible to conduct user studies in the laboratory. For this reason, I decided to run my experiment remotely.

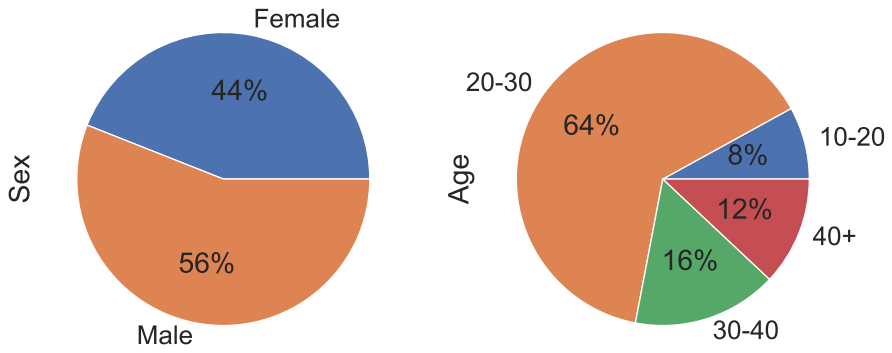


Figure 4.2: Review of sex and age distribution of the participants in my remote experiment.

I collected samples of behavioral interactions with physical objects. The experiment aimed to explore the potential of physical object interaction biometrics for authentication and identification in smart environments. To operate within the lockdown restrictions, I needed the experiment to be performed outside of the lab and without the researcher being there in person (goal G3 of the framework). To achieve this, I had to design the experiment to ensure that every deployment would be consistent (G1) and that the data would be collected in a reproducible manner (G2). Furthermore, this presented me with an opportunity to collect real-world data from apposite users in genuine smart environments (G2).

I ran this experiment from October 2020 to January 2021. During this period, I recruited 25 participants spread across six different households in the United Kingdom and Poland. The recruitment took place through advertisements on social media and a departmental newsletter. In exchange for their participation in my experiments, the participants received Amazon vouchers as compensation.

The youngest participant in the experiment was 11 years old whereas the oldest was 57. To set up this experiment in a realistic environment, I focused on recruiting participants of different ages. My objective was to examine whether physical similarities and familiarity with movements among family members would affect the outcome of the experiment. For this reason, I aimed to include a household consisting of parents and children in my study. Figure 4.2 shows the sex and age distribution among them. Overall, the majority of participants were between the ages of 20 and 30, with 56% male and 44% female.

I delivered ten sensor boards to each household that agreed to participate in the user study. Each sensor board was equipped with an Inertial Measurement Unit (IMU), a microphone, and a magnetic contact sensor, as shown in Figure 4.3. Participants could place any number of such devices on various household objects inside one of the rooms in their home. Figure 4.4 shows examples of sensor board deployments in two households. Although I delivered the same amount of equipment to each participant, some spaces were smaller, so I decided to give them the flexibility to arrange them as they saw fit. Table 4.1 shows the number of participants and devices in each of the households. Participants also received a smartphone with a pre-installed mobile application that allowed them to turn data collection on and off by the sensor boards. More about the architecture of this application and other components can be found in Section 4.4.1.

I prepared a user manual with information on how to proceed with the experiment. I asked each participant to repeat the same sequence of interactions with objects 20 times. However, these 20 trials could be spread over several hours or days depending on users' preferences.

4.4.1 Architecture

To satisfy all the requirements for the safe conduct of a remote experiment, I created an experimentation kit that consists of three main components, including sensor boards, the controller, and the server-side MQTT broker. Each component communicates with the other using MQTT, a lightweight messaging protocol for IoT devices [134]. Figure 4.5 shows the messages exchanged between these components during a round of the experiment. The MQTT broker is the key part as it forwards and stores messages sent to specific topics that all nodes subscribe and publish to. To meet requirement R5, I configured the Eclipse Mosquitto MQTT broker [135] on the server in my research lab to use TLS to secure connections between the broker and the clients. Apart from the MQTT broker, all other components implement the MQTT clients.

Each Raspberry Pi (RPI) board, in conjunction with a Pimoroni[2] module, formed a sensor node. These devices were chosen specifically for their sufficiently high sampling

[2] `https://shop.pimoroni.com/products/breakout-garden-hat-i2c-spi`

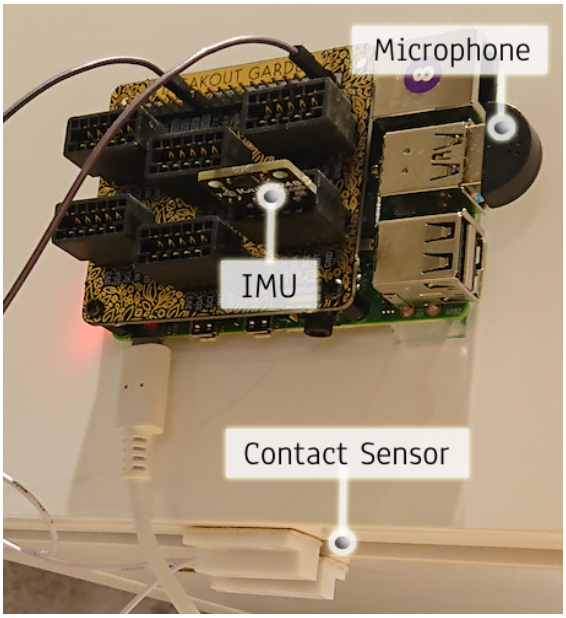


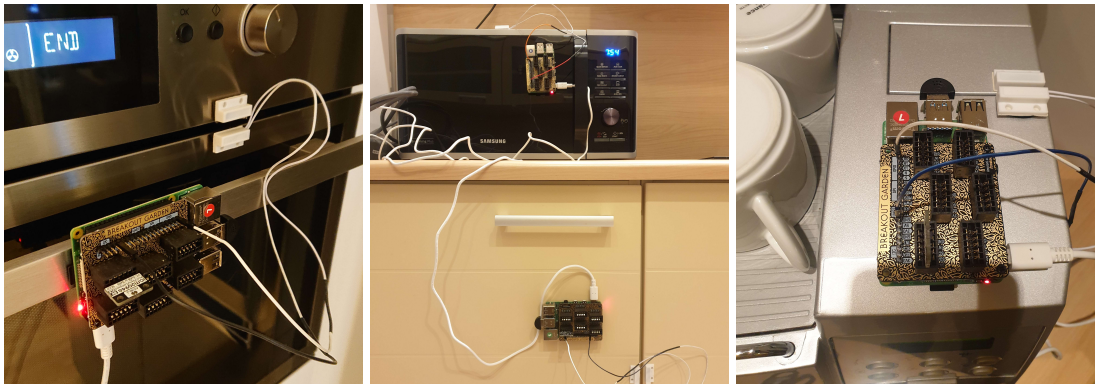
Figure 4.3: Raspberry Pis (RPIs) mimicked smart devices. The participants could freely arrange these in the room. Each RPI board contains an IMU sensor that measures acceleration and has a built-in gyroscope and magnetometer. Each board is equipped with a microphone and a contact sensor.

Household	H1	H2	H3	H4	H5	H6
No. of people	3	2	6	4	3	7
No. of devices	8	10	4	5	4	7

Table 4.1: The number of household members and devices in each household that participated in the remote experiment.

rates and easy-to-use interface. I decided to include the Pimoroni module to facilitate the deployment of the experiment in the participants' environments. Essentially, this module allows various sensors to be easily placed on the RPI board, eliminating the need to solder these components, which satisfies R2. In addition, participants can quickly replace damaged parts with spare plug-and-play sensors included in the experimentation kit. This allows for easy maintenance and therefore complies with R3.

Each RPI board with the Raspbian operating system runs a Python script implemented as a system service that runs on boot and checks network connectivity. The network connection is provided by the controller which acts as a wireless access point. The only steps that participants need to perform are connecting the controller to the local network and enabling a virtual private network (VPN), which is described in the user manual.



(a) oven door **(b)** microwave and drawer **(c)** coffee machine

Figure 4.4: Participants placed sensor boards on various household objects, including an oven door, a microwave, a drawer, and a coffee maker. These are examples of deployments from two households that participated in my remote experiment.

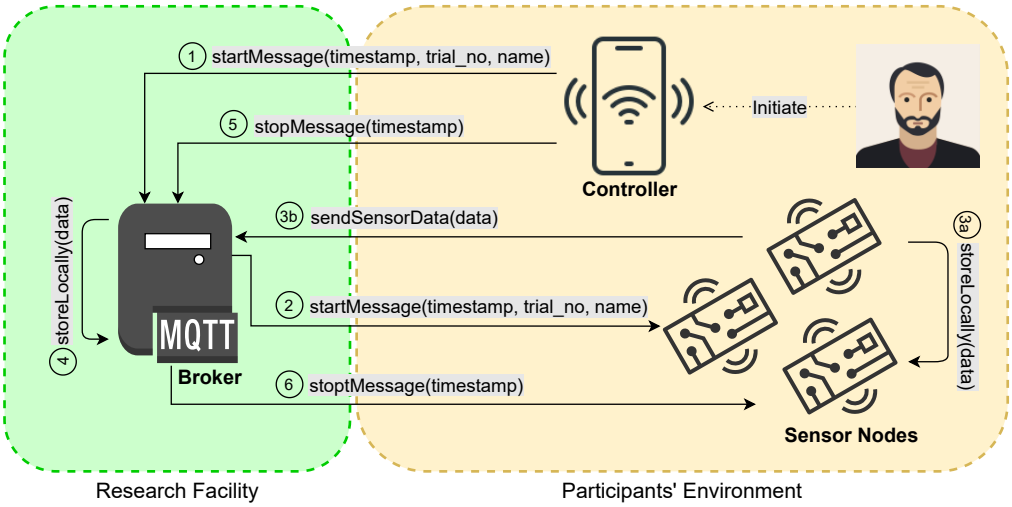


Figure 4.5: An overview of communication between different nodes during the remote experiment. The MQTT broker that relays messages between the mobile application and sensor devices is configured on the server that is placed in a research lab. The backup of the data is stored on the same server. The mobile device and sensors reside in the participants' environments.

Next, the Python script examines whether the sensor components are properly installed and mounted in the Pimoroni module. If the script encounters any problems, it tries to relaunch until the system and its components are fully operational. After connecting to the local network, devices send messages to the server to report their status and identifiers.

Next, the sensor nodes listen to the notifications from the controller. The controller, which in my case is a mobile device with the pre-installed Android application, allows participants to choose the start and end time of each round of the experiment. The sensor



Figure 4.6: The left picture shows the main screen of the controller, where the participant enters his name and the number of deployed sensor nodes. The participant is then redirected to the second screen, which displays the current trial number and the start button. Before this button appears, the application checks whether the sensor nodes are ready for data collection.

nodes do not collect the data until participants have enabled data collection using the controller. The mobile application interface is shown in Figure 4.6. After the participant enters his name and the number of sensor nodes he deployed, the application sends a message containing a timestamp, trial number, and name to the MQTT broker. This message includes a time stamp for synchronization between different components. It is forwarded to all available sensor nodes that respond to the controller to confirm that they are ready to collect the data. The controller will only display a start button if all the deployed sensor nodes are fully operational. After the participants press the start button, these nodes begin logging and sending data to the server. During each round, the sensor nodes write data locally to the RPI boards as a backup. Participants can easily turn off data collection by pressing the end button on the controller.

I delivered sensor nodes, the controller, spare parts, and the user manual to the participants' homes. During the deployment phase, I also provided them with deployment tutorials and videos. The user manual contained information on the configuration of the controller and other electronic components. To satisfy R1, I have also included safety

guidelines for handling electronic components. In addition, the participants were advised to contact me if they encountered any problems. During the experiment, I was able to remotely monitor the status of individual components to check that the data collection was proceeding correctly, which fulfills R4.

Before the experiment, the participants received information sheets containing the description of the experiment and which data would be collected, and a consent form to read and sign.

4.4.2 Survey Results

After conducting my remote experiment, I asked each participant to complete a short questionnaire about their experiences. This questionnaire consisted of ordinal and open-ended questions. My goal was to investigate what challenges participants encountered during the experiment, how helpful the instruction sheets were, and how they rated the overall usability of the experimentation kit. Below are the questions and the results of the survey.

Q1. Were you actively involved in setting up the experiment? The majority of the participants (57%) said that they actively participated in setting up the experiment in their house. Every household had multiple residents, so it was not a requirement that everyone is involved in the setup.

Q2. Have you read the user manual and information sheet before the experiment? I asked this question to investigate whether the participants had used the support materials provided. I found that 29% of them did not read these documents and relied on someone else in the household to explain the experiment. Of those who were primarily involved in setting up the equipment, 88% reported reading the documents. In all households, the sensor nodes were deployed and the experiment was performed correctly.

Q3. What do you think the experiment was about? Every participant answered this question correctly, indicating that they have either read the information sheet that came with the experimentation kit or realized it during the experiment. One of the participants explicitly mentioned that despite the lack of understanding of the subject, he was able to deduce from the course of the experiment what its purpose was.

Q4. How useful was the user manual provided to you? I asked the participants to rate on a scale of 1 to 5 how useful the user manual was, with number 5 indicating the highest usability. None of the participants rated the user manual below 3. The majority, 42%, chose number 4 on the satisfaction scale. The rest of the participants voted equally between 5 and 3.

Q5. Were the instructions in the user manual clear? Next, I wanted to find out if the instructions in the manual were written in a way that was understandable to participants from different professional backgrounds. Again, I used a rating scale from 1 to 5. This time, the participants' votes were split evenly between 5 and 3. These options were voted by 36% of the participants, while 28% of them voted for the number 4.

Q6. How useful was the walk-through video provided to you in the user manual? I included a video in the user manual that demonstrates an example of how the experiment should be performed. I asked the participants to rate the usefulness of this video in the preparation for the experiment: 43% of them rated its usefulness as 3, 29% as 4, and 21% as 5; 7% of participants voted with 1 which means they did not find it helpful.

Q7. How did you find the process of setting up the experiment at your house? The participants' opinions regarding the difficulty of the experiment deployment showed wide discrepancies. In particular, 35% chose a rating of 5, which indicated that setting up this experiment was very easy for them, whereas 42% of the participants voted 4 and 3, and the rest rated it below 2, indicating it was very difficult.

Q8. What challenges did you face while setting up the experiment? 21% of the participants of the experiment answered that they did not encounter any difficulties. The others mentioned several issues they faced during the deployment process, including troubles with attaching the devices to various types of surfaces. The participants reported that the tape they used came off after some time, which meant that they had to repeat the experiment. Another challenge was the positioning of the contact sensor magnets. If the magnets were too far apart, the sensor could not operate properly. The first problem was solved by using a stronger double-sided tape that did not damage the surface. The

second challenge was solved by adding to the user manual the preferred distance between the magnets of the contact sensor.

Q9. Was the mobile application easy to use? On a scale of 1 to 5, 72% of participants gave a rating of 5 for the usability of the mobile application that acted as the controller during the remote experiment. 7% of them rated their experience with the app a 4 and 21% a 3. Overall, nobody reported any issues with the mobile application while experimenting.

Q10. How satisfied were you with the compensation? The majority of the participants (72%) were fully satisfied with their compensation (in vouchers), while 7% of them were not satisfied, explaining that the actual deployment time of the experiment was longer than that mentioned in the information sheet, especially while they had to repeat the experiment when the devices dropped.

Q11. Do you have any suggestions on how to improve the user experience? Finally, I asked participants for feedback on their entire experience with the remote experiment. One person mentioned that better mounting materials should be used for the sensor nodes as they are heavy. Another person suggested introducing a gamification element to make the whole process less tedious. The other participants either mentioned that the experiment was a very interesting experience or had no additional comments.

4.5 Discussion

4.5.1 Research Ethics

Before the commencement of any user study, the experimental methodology has to be reviewed and approved by an appropriate ethics committee. A typical application pipeline is straightforward because it is based on established practices and policies. The researchers design their experiments to adhere to the institutional guidelines as well as common safety rules. These documents have been created based on many years of observation and research experience to facilitate the overall application process. However, often such guidelines do not address the cases in which the experiments have to be conducted outside the research facility, including in participants' households. To better understand the

challenges of conducting experiments remotely, I interviewed a member of the Research Ethics Committee from the Department of Computer Science at the University of Oxford.

Since 2018, this committee has received 121 applications and approved 98 of them. The majority of such applications, namely 68, have been approved after the first round of reviews. The average time to receive comments and detailed feedback from the committee was 12.5 days. While there were individual cases where the overall approval process takes longer, most projects were fully approved in 37.4 days on average.

In 2019, 85 applications were processed. However, in 2020, the committee only received 30 applications due to COVID-19. Because of the imposed restrictions, conducting indoor experiments in research facilities or offices was prohibited. While some user studies easily transitioned to virtual space (*e.g.*, interviews), other types of experiments were postponed. Thus, to tackle this issue, I proposed a non-standard remote experimentation methodology. In my department, I was the first one to propose such an approach and conduct the experiments remotely in the participants' homes. The common practice is to host such experiments on the premises of the university due to several reasons. This allows the researchers to keep the equipment safe and provide necessary maintenance and ensures that the equipment is properly configured so that there are no deviations in terms of settings that may impact the final results. Also, hosting user studies locally allows the researchers to supervise and guide the experimentation process and ensure the safety and privacy of the participants. By conducting experiments in people's private households, the researchers risk accidentally invading their privacy by, for instance, capturing information that they did not wish to share such as their daily habits, schedules, or other types of sensitive information. The committee pointed out that the experiments should be conducted in such a way that will allow the participants to turn off the recording of the data when needed to avoid accidental information leakage. Their other concerns included the physical safety of the participants, so the researchers have to ensure the delivery of instructions on how to safely set up the experiment and take into account other household members, including children and people with disabilities. Moreover, the researchers have to consider their safety. While some of these concerns

are general and applicable to all types of user experimentation, others are specific to the proposed remote experimentation methodology.

I asked about the procedure of applying to conduct experiments off of university grounds. Normally, the committee looks at the time and place of the experiments. Preferably, such studies should be performed within office hours to support the researchers, if necessary. Depending on the user study, it can take place in public spaces, outside office hours, but the researchers have to provide detailed information on the time and location for safety reasons. However, such an approach is not viable in the case of remote experimentation since the participants are the ones who control the time and place of the experiments. Thus, the committee asked me to submit the detailed instruction sheet and the user manual for review as well as to propose a strategy to safeguard. Furthermore, the risk assessment and relevant mitigation strategies have been supplied along with the application. The researchers had to account for the risks related to COVID-19. To suggest appropriate measures, I looked into guidelines from the British National Health Service.

The committee has a comprehensive and formalized experimentation review structure. First, applicants must complete an online application form, answering specific questions about their experimental methodology. Next, they have to submit relevant documents, including consent forms and information sheets for the participants. While the responsibility to teach the researchers how to prepare such documents relies on their supervisors, often the committee provides such guidance throughout the application process. Since my proposed methodology did not belong to any known experimentation category, it had to be reviewed and approved by additional decision-making bodies, including the departmental Safety Committee and the Research Data Management Team, which significantly prolonged the overall approval process. The main reason is that my application could not be easily processed using standard procedures. Typically, if an accident occurs, the educational institution will take responsibility and face specific consequences. If there is a suspicion of negligence caused by the researchers, their actions are assessed under the academic misconduct document. However, during remote experiments the researchers cannot oversee and supervise the entire process; thus, it is challenging to act upon misconduct. Moreover, in addition to considering the legal

obligations, the decision-making bodies had to review the application for compliance with the rules on research grants. Thus, the overall approval process took longer but had drawn attention to important aspects of remote experimentation and helped me to develop a reliable remote experimentation framework.

4.5.2 Lessons Learned

The aim of the experiment was to take realistic samples from participants without the need for direct interaction and to minimize the influence of external factors on the results of the experiment, in line with goals G1 to G3. I designed my user study to meet all framework requirements. Based on feedback from the participants and my observations, I draw the following conclusions with respect to the 5 framework requirements.

R1. My safety planning proved effective and I experienced no safety-related incidents. While there have been no such incidents, I will consider not using too many cables next time to further reduce the risk of potential safety hazards.

R2. All participants understood the experiment (as evidenced by the answers to Q3) and were able to set up the equipment successfully, indicating that the supplied documentation was sufficient. The answers to Q6 and Q7 showed that the video was well received and that most participants were able to set up the equipment without any trouble, while a small number of less technical participants struggled with some parts, indicating that there is still room for improvement.

R3. I experienced some minor problems with adhesive tape and setting up magnets (Q8) in the early iterations of the experiment. These issues were minor inasmuch as they did not necessitate any modifications to the design or ethical considerations of the experiment and were easily resolved in later runs. I also had some damage caused by falling devices (due to the tape) and broken contact sensors (due to shipping, as they are fragile) that impacted some of the samples collected. To address this, a feature that allows reporting the breakages (perhaps integrated into the app) would be useful, especially if done in a way that would absolve the participant of blame and incentivize reporting.

R4. My application (Q9) and data-handling infrastructure were well designed and facilitated the smooth running of the experiment—the initial effort expended on these systems was worth it. My remote monitoring provided me with valuable assurance that the devices had been set up properly.

R5. I handled all data securely and had a full backup available if I needed it. To the best of my knowledge, I did not experience any data leakage or loss.

During my experiment, I controlled key parameters, such as the correct setting and operation of testbed components, and the time synchronization of these components. This was possible due to the use of a controller that gave immediate feedback to the participants as well as remote supervision tools. I was able to collect enough samples to build a comprehensive dataset to conduct the analysis of behavioral biometric traits. However, there were aspects that I could not control. Namely, if the device was damaged or the sensors stopped working during the experiment, some participants did not repeat the trials from before the issue was addressed, which resulted in receiving fewer samples from them. This happened only in two cases and did not affect the result of the experiment, but it should be addressed in the user manual. Factors that I could not control also included accidental outside interference, such as household members coming into the shared space during that experiment or outdoor construction work. Participants were advised to conduct the experiment when no one else was using the shared space—so, when interruptions happened, some of them just repeated the experiment. The construction work was harder to deal with, but I note that this adds realism and can also happen during experiments at a research facility. In general, my remote experiment was successful and the deployment of this framework allowed me to predict and address potential issues early on at the design stage of the experiment.

In summary, this chapter highlights the importance of standardization and interoperability of tools and platforms used in remote experiments. Standardization ensures that the tools are accessible to researchers from various domains and institutions, facilitating collaboration and result-sharing. In addition, I emphasized the need for consistent testbed design to achieve reproducibility of results across multiple experiments. Moreover, I

discussed the importance of data security and privacy, along with clear policies and procedures for data sharing and access. Finally, the chapter described how the framework could enhance participant engagement and the quality of data collected through a user-friendly interface and clear instructions, ensuring participant understanding and motivation to complete the experiment accurately.

The next chapter describes an identification and authentication system that relies on the user's daily interactions with IoT devices. The data collected in accordance with the discussed methodology was used for training and analysis of the system performance. A more detailed description of the setup of this experiment during the study on this system is described in Section 5.4.

5

BeeHIVE: Behavioral Biometric System based on Object Interactions in Smart Environments

Contents

5.1 Motivation 74
5.1.1 Deployment Scenarios 76
5.2 Related Work 76
5.3 System Design 77
5.3.1 Design objectives 78
5.3.2 System Model 79
5.3.3 Adversary Model 80
5.4 Experimental Design 81
5.4.1 Experiment 1 82
5.4.2 Experiment 2 83
5.5 Methods 85
5.5.1 Preprocessing 86
5.5.2 Feature extraction 87
5.5.3 Feature selection 88
5.5.4 Multi-sensor fusion 89
5.5.5 Training 91
5.5.6 Classification tasks 92
5.6 System Evaluation 92
5.6.1 Identification performance 92
5.6.2 Authentication performance 95
5.6.3 Distinctiveness of sensor features 103
5.7 Discussion 105

5.1 Motivation

Based on the preliminary research described in Chapter 3 and extensive studies of user interactions in smart environments in the field of Human Activity Recognition (HAR), I formed a hypothesis that such interactions could be unique to humans and thus be used as a new type of biometric. This chapter analyzes a system that employs this novel biometric using data collected using the aforementioned framework for remote experiments.

In order to motivate my research on the proposed system, I would like to provide some necessary information about smart environments. Projections indicate that by the end of 2022, smart homes will surpass the 84 million mark in Europe alone [136]. As the number of such environments grows, so does the number of devices owned by people in their households or offices. The variety of data collected by such devices not only contributes to providing more customized services but can also be used to protect such environments. These sensors can be leveraged to improve the security of smart environments by providing essential input about user activities.

In many environments, the control over specific devices or financial transactions should only be available to an authorized group of users. For example, smart windows in a child's bedroom should not open when the parent is not present, and the child should not be able to order hundreds of their favorite candy bars using a smart refrigerator. Similarly, not all office workers should have access to a smart printer's history, nor should the visitors in a guesthouse be able to change credentials on smart devices that do not belong to them. Therefore, a smart environment should be able to distinguish who is interacting with these objects at a given moment to decide whether certain actions can be safely executed. While many existing solutions focus on the recognition of specific activities and gestures performed by the users of IoT devices [69–73], in this case, the attribution of those activities to specific household members is desired. Some studies look at identifying users using various on-device sensors [137, 138]. However, these approaches assume the presence of cameras or other specific types of devices in the

environment. This is not only impractical but also does not use the full potential of heterogeneous environments. Moreover, most such systems actively involve users in the identification and authentication process, which negatively impacts their usability.

While there is an inherent need for authentication and identification, smart devices often offer limited interfaces for implementing security measures. This can be mitigated by requiring that the user initiates or approves every transaction through a privileged companion app running on the user's smartphone. However, this can be very cumbersome as the user needs to have their phone at hand, and thus negates many advantages that smart environments offer in the first place. On-device sensors such as microphones, passive infrared (PIR) sensors, and inertial measurement units (IMUs) have been extensively used to recognize different activities performed by users in the area of Human Activity Recognition (HAR) [139]. Prior work has focused on using one type of input data to authenticate users, such as voice, breath, or gait [5–8].

To make attacks more difficult, several systems have been proposed that rely on diverse types of inputs [9–14]. While these approaches are promising, they often do not utilize the full potential of co-located heterogeneous devices in smart environments. To tackle that, I propose the BeeHIVE system that uses co-located and on-device sensor data collected during day-to-day interactions with physical objects to implicitly identify and authenticate users without requiring them to change smart environment hardware or adapt their behavior. This system can be used to complement phone-based authentication mechanisms that require users to explicitly approve transactions through privileged apps. By using BeeHIVE in conjunction with a phone-based authentication mechanism as a fallback, smart environments can become more seamless and unobtrusive for users without sacrificing their security.

To validate my approach, I collected samples of multiple user interactions from six real households with 25 participants in total, using the remote experimentation framework I have described in Chapter 4. Additionally, I conducted a 13-person experiment in a company environment to explore the effectiveness of imitation attacks against my model. The proposed system is assessed in three modes of operation to use (1) features from sensors placed on the object with which the user interacts, (2) features only from

sensors on co-located objects, and (3) features from both on-device and co-located sensors. Overall, the outcome of my analysis proves that the system achieves desirable security properties, regardless of the number of smart office users or the environment configuration.

5.1.1 Deployment Scenarios

The need to identify and authenticate users arises because physical access to smart objects does not imply authorization to use them. I consider scenarios in which children or visitors may abuse the trust of their parents or hosts to initiate sensitive operations through smart devices that are unwanted by the owners of said devices. These operations can include making payments to other people, ordering goods online, changing the configuration of smart devices, or accessing sensitive information stored on these devices. For example, a child might want to exploit the restocking mechanism of the fridge to order their favorite sweets while a visitor might unintentionally or intentionally access the viewing history of the smart TV and learn intimate details about their hosts.

Other possible settings include offices where smart devices are accessible to staff and visitors alike. Often these smart devices allow users to complete administrative tasks through them, such as reordering supplies or accessing the print job history of smart printers. But access to this functionality should be restricted to authorized personnel. In these cases, an implicit identification of the person executing these tasks as done by my system can avoid cumbersome external authentication methods.

5.2 Related Work

Closest to `BeeHIVE` is SenseTribute [82], which performs occupant identification by extracting signals from physical interactions using two on-device sensors—accelerometers and gyroscopes. Its main objective is to attribute physical activities to specific users. To cluster such activities, SenseTribute uses supervised and unsupervised learning techniques, and segments and ensembles multiple activities.

There is a palpable risk in real-world smart environments that users will attempt to execute actions that they are not authorized. This requires means for not just identification, but also authentication. Therefore—in contrast to SenseTribute, which focuses on user

identification—the main objective of BeeHIVE is user authentication, for which I conduct a more extensive experiment evaluating various types of active attacks. In office and home environments, it is easy for anyone to observe interactions made by authorized users, and it is natural that, for example, kids may seek to imitate their parents. Going beyond previous work, I, therefore, evaluate the robustness of my system against mimicry attacks based on real-time observation or video recordings. Furthermore, SenseTribute expects all objects to be equipped with sensors. However, this is not always a realistic assumption, as sensors are often deployed only near (but not on) interaction points. Thus, I propose a system that uses nearby sensors present in co-located IoT devices to identify and authenticate user interactions.

In summary, BeeHIVE extends SenseTribute in several ways by considering more sensor modalities, several types of smart environments instead of just one, and an increased number of users. More importantly, the proposed system can employ co-located sensors, which are not located directly on objects, to fingerprint users. This enables not only significantly improved identification performance in comparison to SenseTribute but also makes it easier to deploy in existing smart homes or offices, as it uses the natural locations of smart devices without restrictions on their placement. It also enables identifying interactions with objects that do not have any sensors of their own. Lastly, my work uses the novel remote experimentation methodology introduced in Chapter 4 to collect more realistic biometric data from experiment participants.

5.3 System Design

The heterogeneous nature of smart devices makes it possible to sample different types of user interactions. The main purpose of this work is to show that such interactions with various objects in smart environments are distinctive and can be used to profile users. The expansion of smart devices, and hence smart environments, will soon make such methods necessary to quickly authorize certain activities, including payments or management of smart devices. Furthermore, user identification is necessary to enable user-specific configurations, track access or usage patterns (to improve user experience), or implement parental controls. Figure 5.1 shows an overview of the system design. The

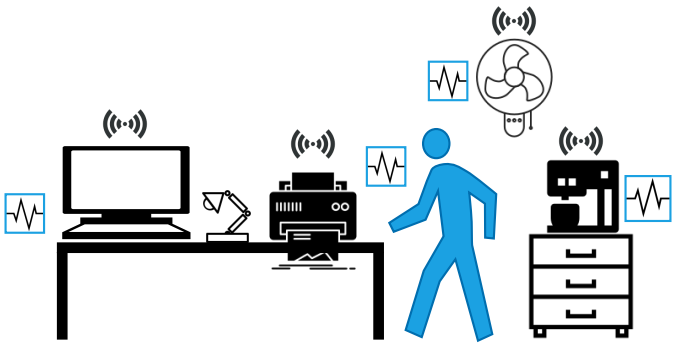


Figure 5.1: An overview of the BeeHIVE system. As the user interacts with the printer, sensors embedded in smart objects surrounding the user and the printer record these interactions. Physical signals generated from the user's movements are picked up by sensors such as accelerometers, pressure sensors, and microphones, and are used to profile them. The system authenticates the user before allowing them to perform certain actions, such as payments.

proposed BeeHIVE system is meant to complement existing app-based authentication mechanisms used to secure current smart home platforms. This system authenticates the user through their interactions with the smart environment and only requires the user to approve transactions through the app as a fallback if it cannot authenticate the user with confidence itself. In this way, BeeHIVE can be used to reduce the reliance on these app-based authentication mechanisms without compromising the security of the smart home platform.

5.3.1 Design objectives

In order to inform the system design and evaluation methodology, I define the following design objectives:

O1. **Unobtrusiveness.** The system should not require users to perform explicit physical actions for the purpose of authentication nor require them to modify their usual behavior.

O2. **Low false accept rate.** As the system is designed to be used alongside app-based authentication, it should prioritize low false accept rates to avoid significantly weakening the security of the overall smart environment system.

O3. **Low friction.** The system should provide a seamless experience to the user wherever possible. This means that false reject rates should be kept low to reduce the need of falling back on the usual app-based authentication of the underlying smart environment platform. However, this should not come at the cost of higher false accept rates.

O4. **No restrictions on sensor placement.** The system should use data from existing sensors without making restrictions on their placement or orientation. This ensures that the system can be applied to existing deployments purely through software. In addition, the system should not require sensors on each object but instead use sensors on other nearby devices.

O5. **Robustness to imitation attacks.** Due to the ease of observation in home environments, the system's error rates should not increase significantly even when subjected to imitation attacks.

5.3.2 System Model

BeeHIVE applies to smart environments where objects such as fridges or cupboards are augmented by smart devices that monitor their state and provide access to enhanced functionality. People naturally interact with many of these smart objects during their daily activities. For example, the user may first open the dishwasher to start unloading it. Then they open various cabinets and drawers to put away clean dishes. These interactions have physical effects on the environment that can be picked up by nearby sensors, and—as each type of interaction is performed uniquely by different people—they can provide a behavioral fingerprint that allows the system to distinguish and identify different members of a smart environment. Moreover, sequences of these interactions can be used to further increase system performance. In this research work, I focus on a single user being identified at a time and do not consider multiple concurrent users as a simplification. BeeHIVE does not require the object that is part of an interaction to be equipped with sensors itself. In fact, it is often beneficial for the system to consider sensors on other close-by objects. In this work, I, therefore, consider three possible deployment configurations:

- ON-OBJECT, where sensors are mounted directly on the object
- OFF-OBJECT, where only co-located sensor data are considered
- COMBINED, which uses sensor data of both the device on the object as well as from co-located devices

It is important to note that in this system a failed authentication does not mean that the user is barred from making transactions. Instead, they can simply not benefit from the seamless authentication provided by `BeeHIVE` and are required to use their phone to approve the requested transaction.

5.3.3 Adversary Model

As the `BeeHIVE` system identifies the person that interacted with physical objects, I consider an adversary (Adv) to have physical access to the environment and the devices therein. Due to this assumption, Adv could be relatively benign (*e.g.*, a child wanting to access restricted TV channels or order their favorite food using the smart fridge). In any case, Adv's main objective is to convince the smart environment that they are a legitimate user (U_L). Such a misclassification can result in permitting Adv to execute on-device financial transactions or any other types of sensitive operations on behalf of U_L. As a result, I exclude more sophisticated attacks such as modifying sensor firmware (*e.g.*, connecting to the debug port to flash the device firmware), tampering with network communication, spoofing sensor data, or poisoning the training phase. I also assume that smart devices and the user's smartphone are not compromised; thus, they can be considered reliable data sources. Based on these assumptions, I also exclude the possibility of Adv interrupting the training phase, which could result in the generation of incorrect biometric signatures of authorized users.

Instead, to achieve their goal, Adv may attempt to mimic the behavior of U_L to generate a matching biometric fingerprint. Successful mimicry attacks on various biometric systems have been previously demonstrated [140]. In my research work, I consider three types of such attacks:

1. **Zero-effort** attackers who interact with the environment naturally without attempting to change their behavior

2. **In-person** attacks in which Adv can observe legitimate users interacting with IoT devices in person

3. **Video-based** attacks in which Adv possesses a video recording of the user interacting with the IoT devices in a smart environment

While in-person attacks give Adv a possibility to inspect U_L's interactions more closely and potentially capture more details, recordings can provide additional time to learn U_L's behavior. For zero-effort attacks, I assume Adv will interact with devices in the normal manner and be successful if they are incorrectly identified as an authorized user. It is worth noting that my accuracy metric is stricter than this, as a child being misidentified as another unauthorized child is considered a wrong classification but would not lead to incorrectly denying or granting access.

5.4 Experimental Design

To evaluate both the identification and authentication performance of the BeeHIVE system, I conducted two experiments. The first one involved gathering data in real-world homes of various household sizes to study the uniqueness of physical interactions for user identification in multi-user environments. The deployment of this experiment was referenced in Section 4.4 of Chapter 4. The second experiment followed the same remote experimentation methodology, however, was conducted on the premises of a hotel company to study mimicry attacks. The participants in both experiments received similar instructions listed in Appendix A.

The sensors used in both experiments are magnetic contact switches (which can detect open/close events similar to door/window contact sensors), USB microphones (which were only used to measure sound pressure levels), as well as ICM20948 IMUs (which consist of an accelerometer, a gyroscope, and a magnetometer). These sensors were mounted on and controlled through ten Raspberry Pis (shown in Figure 4.3). I used these

custom Raspberry Pi sensor boards because raw sensor data in commercial smart devices are typically inaccessible to developers. The sensor boards were deployed in places where equivalent smart devices are already common in most smart environments, such as home appliances or kitchen furniture (shown in Figure 4.4). The ground truth for events related to interactions with smart objects—such as the opening of a kitchen drawer or the closing of a fridge door—is provided by the aforementioned magnetic contact switches.

5.4.1 Experiment 1

To study the use of smart home interactions for user identification, I collected data using types of sensors that can typically be found in modern smart homes. Following Covid-19 regulations, this experiment was conducted remotely in the kitchens of the participants.

When recruiting participants for this experiment, my aim was to ensure the inclusion of individuals from diverse age groups, including the elderly. Excluding participants who are elderly or have disabilities from biometric experiments will lead to a limited sample size that is not representative of the entire population. This could introduce biases into the results, which may not accurately reflect the true accuracy of the proposed system.

The number of used devices varied between households depending on the size of the kitchen, ranging from six to ten devices. Each of these devices corresponded to one object interaction (*e.g.*, opening and closing the fridge door). The object decomposition per household is shown in Table 4.1.

The participants in each household chose these interactions themselves based on their kitchen layout. These layouts are shown in Appendix B. A breakdown of where the devices were installed for each household is given in Table 5.1. For each of these interactions, one Raspberry Pi device is fitted to the corresponding monitored point of interaction. The only exceptions to this are the devices mounted on the floor in household H1. These devices do not provide any ground truth for device interactions but are instead used to give auxiliary sensor data to other interactions based on the gait characteristics of the user as they move around the room.

Table 5.1: Device location and interaction type breakdown. Numbers in parentheses show the number of related devices.

Device location	H1	H2	H3	H4	H5	H6
Fridge/Freezer	✓(2)	✓(2)	✓(2)			
Cupboard	✓(2)	✓(3)	✓(2)	✓(2)	✓(2)	✓(3)
Floor	✓(2)					
Pull-out drawer		✓(1)		✓(1)		✓(1)
Microwave door	✓(1)	✓(1)		✓(1)	✓(1)	
Oven door		✓(1)		✓(1)		✓(1)
Dishwasher	✓(1)	✓(1)			✓(1)	✓(1)
Washing machine		✓(1)				
Bread box						✓(1)

During every run of the experiment, participants perform each interaction exactly once. Each participant completes 20 runs of the experiment, resulting in a total of 3000 runs for 25 participants across the six households.

5.4.2 Experiment 2

In order to evaluate the feasibility of authenticating users seamlessly based on their interactions with smart devices, I conducted an experiment in a smart office environment with thirteen employees of the same company who deployed a total of ten Raspberry Pis equipped with the aforementioned sensors to collect the data. This experiment is further used to study adversaries that attempt to copy the behavior of the legitimate user to execute mimicry attacks.

An overview of the deployment and the room layout is shown in Figure 5.2. As object interactions, I consider in this experiment: 4 cupboards, 1 mini oven, 1 pull-out drawer, 1 microwave, and 1 coffee machine. Apart from the coffee machine, all of these interactions involve the opening and closing of the doors of the interaction point. To get the ground truth for the coffee machine interaction, the user first opens a lid on top of the coffee machine which is outfitted with a magnetic contact switch. The user then

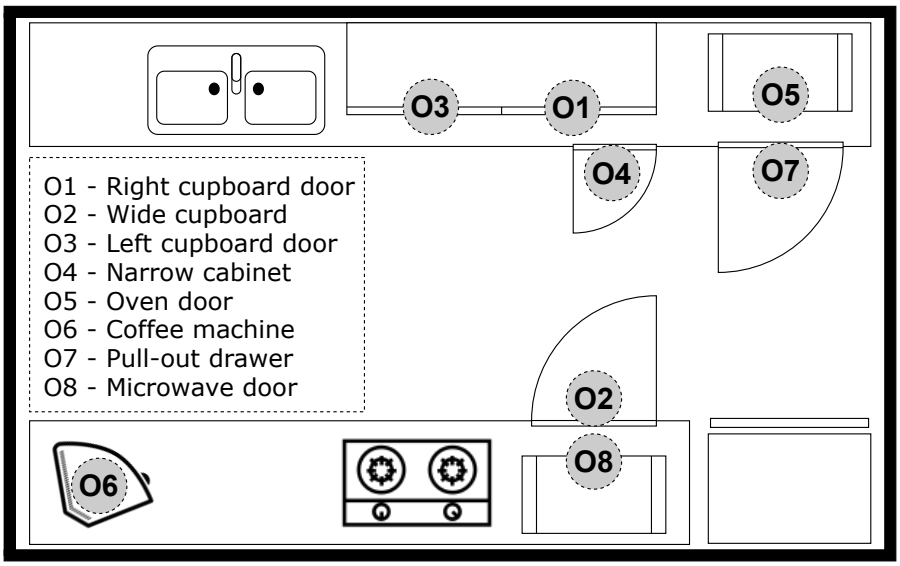


Figure 5.2: A simplified layout of the room and the arrangement of the objects $O1 - O8$ the participants interacted with during the experiment.

proceeds with pressing buttons on the coffee machine, before they end the interaction by closing the lid on top of the machine again.

Each of the participants performs 20 runs of interactions. Then, one of the participants is randomly chosen as the legitimate user and victim of the attack. The rest of the participants are split into two groups of six attackers who can observe the user's interactions with the smart environment. The first group can only observe the victim in-person, whereas the second group has access to video recordings of previous object interactions which they can study in their own time.

The participants from both groups of attackers then execute the same interactions as the victim, carefully trying to mimic the victim's behavior. The attackers from the first group (*i.e.*, the in-person group) have to perform this attack on the same day when the observation took place. The participants in the second attack group can watch video recordings of the victim from different angles overnight and only have to execute the attack on the following day.

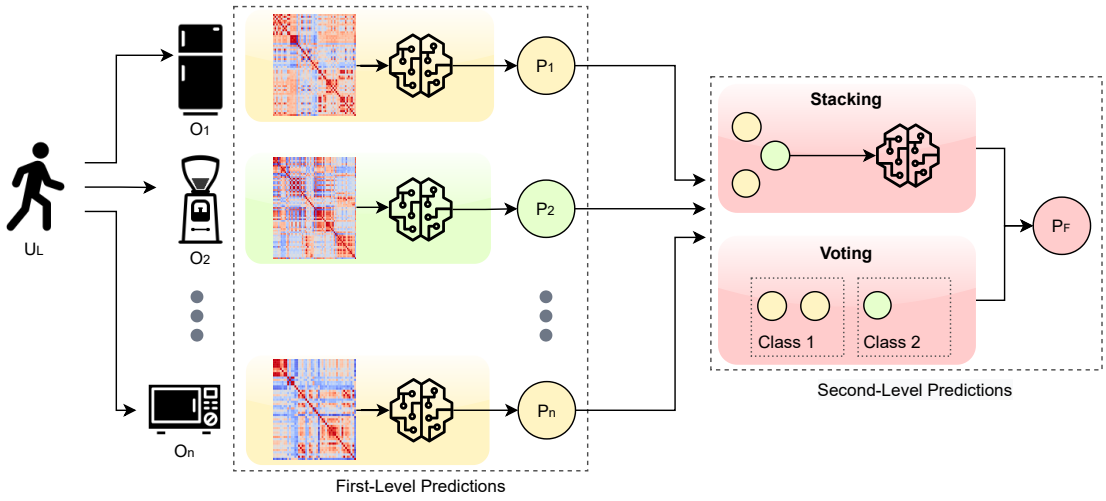
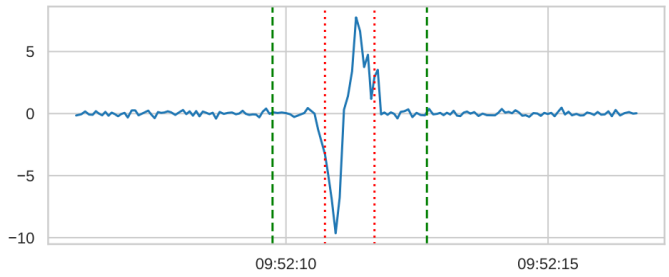


Figure 5.3: The diagram provides an overview of the processing pipeline of a multi-sensor fusion system. The system extracts relevant features from U_L's interactions with objects O_1 to O_n and supplies them to their base classifiers. Then, the first-level predictions P_1 to P_n are fed into a meta-classifier (*i.e.*, a voting or stacking classifier) that computes the final prediction P_F.

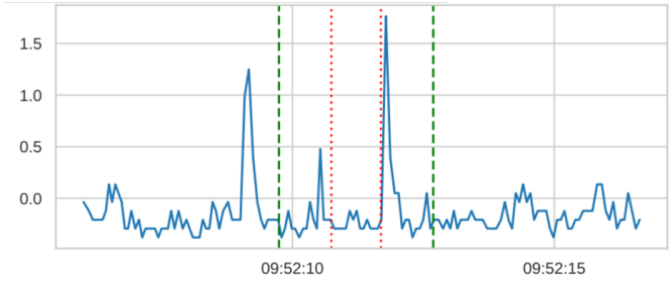
5.5 Methods

I formally define a task T as a physical interaction initiated by the user U_L with an object O. Each task can be modeled as a time series $T = \{X_1, X_2, \ldots, X_z\}$, which is constructed from the data collected by on-device sensors, including microphones, accelerometers, gyroscopes, and magnetometers. The variable X_t represents a physical signal generated by the user while they interact with the smart object at time t in form of a vector of sensor values. Depending on the combination of sensors on the devices, they can collect diverse inputs. For example, a smart refrigerator equipped with Inertial Measurement Units (IMUs) can collect acceleration values as vectors of $\langle a_x, a_y, a_z \rangle$ when the user opens or closes its door. However, a smart coffee machine may be equipped with both an IMU and a microphone, which results in collecting more input data.

Figure 5.3 presents the system overview and explains its processing pipeline. Base-learners are weak classifiers that are combined to form an ensemble to facilitate the decision-making process. When the user performs a sequence of tasks T_1 to T_n on several smart objects, the system extracts the features for these tasks from on-object sensors as well as sensors in proximity. Next, the features become an input to the n base-learners corresponding to those tasks—resulting in predictions P_1 to P_n. These



(a) The interaction of the user U_1 with the narrow cabinet as read by the x axis of a gyroscope.



(b) The same interaction of the user U_1 with the narrow cabinet picked up by the microphone of the co-located wide cupboard $O2$.

Figure 5.4: As users interact with smart devices, signals from on-device sensors are collected and processed by the system. Signals (a) and (b) are generated by the participant U_1 that has interacted with the smart object $O4$. $O2$ picked up additional input from the same interaction with object $O4$ as they were co-located. More specifically, these plots illustrate the movements on the x axis of the gyroscope (a) of $O4$ and RMS values (b) picked up by the microphone of $O2$. Red dotted lines indicate the start (t_0) and end (t_1) of the T task while the green dashed lines denote $\pm$ 1-second windows.

predictions can either indicate a probability that an observed sample belongs to a certain class or a concrete label from the set of labels $L = L_1, L_2, \ldots, L_m$, depending on the framework configuration. Finally, the meta-learner gathers all predictions made by all the base learners and decides on the final prediction P_F in the second-level prediction layer. This way, a smart environment can benefit from the heterogeneous character of smart devices and their built-in sensors by performing a decision-level fusion to improve classification accuracy.

5.5.1 Preprocessing

The multitude of U_L's interactions in a smart environment translates into physical signals that are received by the sensors of smart devices. Figure 5.4 presents the sensor readings when U_1 interacts with the narrow cabinet during the experiment. While (a) shows the

signal that the gyroscope sensor of the cabinet has captured, (b) reveals what has been registered by a co-located sensor. Co-located sensors are all sensors in proximity to an object that can capture physical signals originating from interaction with this object. The microphone on the wide cupboard recorded two events — opening and closing the door of the cabinet. These movements are part of the task T performed on smart object O. The start and end of T are time-stamped by the contact sensors and denoted as t_0 and t_1 respectively (marked with red dotted lines in Figure 5.4). The signals from T are segmented by the values of $t_0 - 1$ and $t_1 + 1$ before proceeding to the feature extraction phase. The time-series signals are converted to values characteristic of h sensor types. As a result, for each T, a set of corresponding matrices $S_1, S_2, ..., S_h$ exists that contains vectors of length m of different sensor values X_i between t_0 and t_1. Thus, with $j \in 1, 2, \ldots, h$ and the sensor component c_a where $a \in 1, 2, \ldots, m$, I refer to a single matrix as S_j with columns $c_1, c_2 \ldots, c_m$. The number of columns m for S_j is determined by the sensor components (*e.g.*, three axes of a gyroscope sensor S_1 gives $m_1 = 3$). For a smart object with two corresponding sensors, two such matrices will be generated. The variable Z_{t_i} represents a component-specific input value X_{t_i} for a column c_a generated by a physical signal received by sensor S_j. These matrices are then passed as input to the feature extraction function.

5.5.2 Feature extraction

For each physical interaction with an object O_i, the system extracts h matrices with $g = \sum_{j=1}^{h} m$ columns of time series data segments of particular sensor components of this object and co-located objects. Based on these columns, the features are extracted. More formally, for each smart object O_i the system extracts a set of features F_{O_i} such that $F_{O_i} = \{F_r : 0 < r \leq R * g\}$, where:

$$F_r = \{F_q([S_j]_a) : 1 \leq q \leq D\} \quad for \quad [S_j]_a = \begin{pmatrix} Z_{t_0} \\ \vdots \\ Z_{t_1} \end{pmatrix} \tag{5.1}$$

Table 5.2: The decomposition of features that smart devices extract from users' physical interactions.

Types	Features
Time-domain features	Min, Max, Mean, Median, Std, Var, Kurtosis, Skewness, Shape factor, Absolute energy Mean of central approx. of 2^{nd} derivative, Mean/Sum of abs change, Peaks
Frequency-domain features	Fourier entropy

F_r represents a set of feature values extracted from a component a (*e.g.*, an x-axis of an accelerometer) in S_j of length D. R denotes the total number of different functions that extract features.

The features retrieved from physical signals are listed in Table 5.2. The statistical functions are computed from each column of S_j, which contains sensor values extracted between $t_0 - 1$ and $t_1 + 1$. The system adds windows of a second to account for signals that originate from the starting and ending movements. These features are categorized into two groups: time-domain features and frequency-domain features. The majority of the extracted features originate from the time domain because such features are typically well-suited for systems that process large volumes of data due to their low computational complexity. These features help to analyze the biomechanical effect of a given interaction on physical signals and identify characteristics of movements [141]. For microphone data, the system extracts sound pressure levels (SPLs) instead of actual audio recordings. Thus, statistical functions are applied to SPL values.

5.5.3 Feature selection

For each smart object, the system selects a subset of extracted features using a filtering method. This method focuses on verifying whether features are relevant by analyzing their association with the target variable. The univariate feature selection method used in this work relies on statistical tests to investigate the relationship between variables. The features selected from various sensors are aggregated and become an input to an object-specific base-classifier. This is a necessary step in my pipeline to improve the model performance as well as reduce computational complexity.

Mutual information

Mutual information (MI) is used to examine the distinctiveness of a set of features and to test the null hypothesis (H_0) that negates the existence of a relationship between a feature and an associated target variable. This method can capture statistical dependencies between variables, explaining whether one variable can provide relevant information about the other one [142]. By accepting H_0, the system assumes that the extracted feature is not relevant, indicating that it is independent of the target variable. On the other hand, rejecting H_0 suggests that the variables can be dependent, so the feature should be considered relevant. In practice, the null hypothesis is rejected or accepted after examining the resulting non-negative MI scores. The higher the score, the more significant the feature may be. The zero score indicates independence between the variables. As there can be many relevant features, the system has been configured to choose only the top 20 of such features ranked by the highest scores for each object.

5.5.4 Multi-sensor fusion

As stated in Section 5.3.2, I explore and evaluate the potential of heterogeneous smart environments to authenticate and identify users while they carry out their daily activities. Every node in a smart environment extracts different sets of characteristics from user interactions due to their placement, purpose, and composition of built-in sensors. Various fusion approaches exist that can boost the detection accuracy and system effectiveness in multi-sensor environments [68]. Among these fusion techniques, I focus on *decision-level methods* which allow the introduction of multiple classifiers, *base-learners*, that independently undertake a classification task. This gives a certain degree of autonomy to individual base learners trained on specific smart object interactions. Moreover, such an approach allows my system to select the most effective feature sets, classifiers, and their hyper-parameters for each of the base learners. As shown in Figure 5.3, after each first-level base-classifier makes a prediction, the second-level meta-classifier determines the outcome. The efficiency and effectiveness of various fusion techniques at the decision level have been extensively studied in the area of Human Activity Recognition (HAR) [68]. While my focus is on user identification and authentication, I hypothesize that similar

approaches can be just as effective for my system. As such, I compare two ensemble learning techniques that use fundamentally different classification methods but show promise for good performance in multi-user smart environment scenarios.

Stacking

A meta-learner is trained using labels obtained from the first-layer base-learners, as its features [143]. I chose stacking as a method of linking heterogeneous classifiers since it typically achieves high accuracy and introduces less variance than other approaches. Combining a multitude of smart objects and their classifiers can be helpful because some of these interactions can classify certain users better than others. The optimal parameters of the meta-classifier are determined during the training phase by using cross-validation on the training dataset to avoid overfitting. Based on the combinations of predictions that the meta-learner receives from the base classifiers, it computes the most accurate label. Stacking allows combining various classifiers (*e.g.*, k-Nearest Neighbours, Random Forests, Decision Trees, etc.) using different sets of features for each. For my system, the biggest advantage of this approach is that the meta-classifier learns which object interactions predict labels more accurately. For instance, after a training phase, a meta-classifier learned that tasks performed on O_1 or O_3 are more effective in recognizing U_L than O_2. Therefore, it will account for it in the future while making predictions. The varying classification effectiveness of individual object interactions and their sensors was a major factor in deciding whether to include the ensemble learning methods in my system.

Voting

Voting is another ensemble learning method. In comparison to stacking, this technique does not require a separate machine learning model to make final predictions. Instead, it uses the deterministic hard voting algorithm to compute the result. In my scenario, voting simply indicates that the most reported class label from the set of predictions will be selected as the outcome P_F.

$$P_F = \text{mode}(P) \tag{5.2}$$

Table 5.3: Search space for Random Forest (RF) hyperparameters. As each base classifier chooses its parameters, the optimal values given here are the most commonly chosen ones.

Parameter	Search space	Optimal value
Number of estimators	10, 50, 100, 200	100
Tree depth	2, 4, 5, 6, 7, 8	7
Number of features	$\sqrt{N_F}$, $\log N_F$	$\sqrt{N_F}$

Table 5.4: Search space for Support-Vector Machine (SVM) hyperparameters. As each base classifier chooses its parameters, the optimal values given here are the most commonly chosen ones.

Parameter	Search space	Optimal value
C	0.1, 1, 10, 100	0.1
γ	1., 0.1, 0.01, 0.001	0.01
Kernel function	linear, polynomial, rbf, sigmoid	rbf

For example, let's assume that there is a set of predictions $P = \{P_1, P_2, P_3\}$ computed for three smart object interactions. For the first object, a smart fridge, the system computed P_1, which resulted in the label L_1 belonging to U_L. Then, a smart microwave did not recognize U_L. But for the coffee machine, the system calculated P_3 that again pointed to L_1. Thus, the final prediction would result in L_1. I decided to assign uniform weights to all base classifiers and test how different combinations of such classifiers affect the performance of the ensemble system.

5.5.5 Training

During the system's training phase, each user labels their interactions with their identity after they have authenticated themselves (*e.g.*, via a setup PIN) to avoid mislabeled data and training data poisoning. These interactions during the training phase have to be performed individually (one user at a time). In my experiments, this labeling was implemented through a smartphone app (see Figure 4.6). Each feature vector generated for an object interaction during training is then labeled with the user identity provided. To evaluate the system, I split the total dataset into training and test data using 10-fold cross-validation. This ensures each sample (object interaction) is used for testing exactly

once. The training data generated by this process is then used to train the classifiers as described in the following subsection.

5.5.6 Classification tasks

The system performs a supervised learning task that focuses on multi-class classification with the dataset collected during Experiment 1 to evaluate identification performance and binary classification with the data gathered during Experiment 2 to assess authentication performance. The features are extracted for each base classifier in three different system configurations, namely ON-OBJECT, OFF-OBJECT, and COMBINED. These base classifiers create the first-level predictions P_1 to P_n and on their basis, the meta-classifier generates the second-level prediction P_F. Since both the Support Vector Machine (SVM) and Random Forest (RF) models tend to outperform other classifiers in HAR tasks [144], I chose them as the base classifiers in the task. For each object interaction classifier, a grid search is performed to find the optimal set of hyper-parameters. More details can be found in Tables 5.3 and 5.4. For all SVM-based base learners, the selected features are first standardized. The resulting classification accuracy is averaged over different folds and used to select the best models. For stacking, as the second layer model, I chose Logistic Regression due to its simplicity and ease of interpretation [145].

5.6 System Evaluation

5.6.1 Identification performance

First, I evaluated the identification performance of the `BeeHIVE` system and compared it to SenseTribute [82]. For each household, I obtained a different dataset. The number of samples in each dataset varies based on the number of household members that participated in Experiment 1.

Table 5.5 presents the averaged accuracy scores in different households over individual objects. The results are shown for ON-OBJECT, OFF-OBJECT, and COMBINED configurations. Apart from two households, the COMBINED configuration exhibits the best performance and ON-OBJECT the lowest. The ON-OBJECT configuration is

Table 5.5: Comparison of average accuracy results from individual objects between different households given ON-OBJECT, OFF-OBJECT, and COMBINED configurations. The numbers in parentheses indicate the number of household members in each household.

	H1	H2	H3	H4	H5	H6
No. members	(3)	(2)	(6)	(4)	(3)	(7)
ON-OBJECT	0.97	0.99	0.93	0.92	0.97	0.88
OFF-OBJECT	0.96	1.00	0.96	0.89	0.98	0.91
COMBINED	0.95	1.00	0.99	0.90	0.98	0.93

particularly low for $H6$, which can be explained by the placement of some sensors on the semi-automated cupboard, which resulted in the loss of some vital interaction data for this object. In comparison to SenseTribute, which achieves only 74% average on-object accuracy in a 5-person environment, the BeeHIVE system achieves a higher score of at least 88% in more populated environments, resulting in a 14% improvement in identification performance over the SenseTribute system. Interestingly, relying solely on co-located sensors (OFF-OBJECT) to capture the unique characteristics of the interaction is sufficient to achieve high identification results. Overall, I observed that including the data from both on-device and co-located sensors benefits the system the most. However, I also noted that this depends on the physical arrangement of the objects in the house as well as their spacing. Unlike in other households, there were greater distances between objects in $H1$ and $H4$. This confirms my hypothesis that the physical setup will determine which system configuration is better for a given smart environment. The main benefit of BeeHIVE is that it is able to leverage sensors of co-located devices. If co-located sensors are taken into account, the minimal identification accuracy in $H6$ can be improved by 3% and 5% for OFF-OBJECT and COMBINED settings, respectively. Aside from significantly improving the identification results compared to the scenario used by SenseTribute—where only sensors on the device that is being interacted with are considered—this also makes my system more easily deployable, as it does not require that the sensors used for identification are located on the device in question.

Table 5.6 shows the accuracy scores for nine types of common household objects in the COMBINED configuration. This provides a more detailed view of the performance based on various kinds of interactions with objects across all households. Since the experiments took place in the participants' houses and each had a different set of

Table 5.6: Device interaction type and accuracy breakdown for individual devices using the COMBINED configuration.

Device location	H1	H2	H3	H4	H5	H6
Cupboard	0.96	1.00	0.99	0.95	0.98	0.91
Microwave door	0.89	1.00	-	0.80	0.98	-
Dishwasher	0.96	1.00	-	-	0.98	0.95
Fridge/Freezer	0.97	1.00	0.98	-	-	-
Pull-out drawer	-	1.00	-	0.90	-	0.96
Oven door	-	1.00		0.96	-	0.94
Floor	0.96	-	-	-	-	-
Washing machine	-	1.00	-	-	-	-
Bread box	-	-	-	-	-	0.92

Table 5.7: Comparison of accuracy results between different households. Second-level predictions are generated using all the available objects in the household in the ON-OBJECT configuration and the Random Forest meta classifier. The numbers in parentheses indicate the number of household members in each household.

	H1	H2	H3	H4	H5	H6
No. members	(3)	(2)	(6)	(4)	(3)	(7)
Voting (V_{OT})	1.00	1.00	0.96	0.99	1.00	0.98
Stacking (S_{TC})	1.00	1.00	0.98	1.00	1.00	1.00

objects, I could not always ensure that the same types would be used. Therefore, the classification performance for each object type is calculated with different sample sizes, depending on the household. The interactions with microwave doors appear to have the lowest classification performance, but still, achieve an average accuracy of 80% in a 4-person household. Typically, the COMBINED configuration improves the identification performance of objects with a low accuracy of interaction classification; however, as I mentioned above, in $H1$ and $H4$, the objects were far apart. Regardless of the size of the household, the average accuracy scores for interactions with other objects exceed 90%. In general, I observed that the interactions with the objects considered in Experiment 1 are quite distinctive, which explains the good performance of the meta-classifiers.

The comparison between the identification performance of voting (V_{OT}) and stacking (S_{TC}) meta-classifiers across different households is summarized in Table 5.7. The accuracy scores are calculated for ensembles of all objects in the ON-OBJECT configuration. I decided to present the worst-performing configuration to highlight the benefit of including more interactions. The table reveals that S_{TC} performs better. In this

setting, the system learns which objects have better classification performance and relies on their predictions, rather than considering all. Unlike stacking, voting assigns equal weights to all base classifiers; thus, they equally contribute to the final prediction. This could be further improved by assigning smaller weights to objects that exhibit worse performance. As expected, the performance of the two meta-classifiers begins to diverge as the number of household members increases. For instance, in households with 2–4 members, there is almost no difference in terms of the performance of the two meta-classifiers. However, for 6 and 7 people, the difference between the performance of V_{OT} and S_{TC} is 2%. Overall, the stacking meta-classifier outperforms the voting meta-classifier in terms of identification performance. While this may also depend on the physical and behavioral differences between particular users (*e.g.*, family members versus a group of students), I expect that in other households this would yield similar results. In comparison, SenseTribute reaches a maximal identification accuracy of 96% in their single five-person household setup when multiple objects are considered.

Figure 5.5 shows the average performance of different ensembles of unique objects (inc. 95% confidence intervals) in $H6$ in the COMBINED configuration for stacking and voting meta-classifiers. I focus on this household as it was the largest one in my experiment. The identification performance of the system, regardless of the selected meta-classifier, gradually improves when it considers more objects. Overall, S_{TC} achieves higher accuracy scores than V_{OT}. This difference is especially evident when considering two object interactions. The system that uses a stacking meta-classifier offers a 5% improvement in the identification performance. The reason is, as explained earlier, that V_{OT} assigns equal weights to all base learners. However, the gap between the ensembles of different sizes slowly decreases as more objects are included. Thus, the system can identify the users with high accuracy considering fewer object interactions.

5.6.2 Authentication performance

In Experiment 2, I focused on analyzing the system performance against three types of attacks. The first part of the dataset contains the samples from the victim as well as zero-effort attack samples from each of the remaining twelve participants. This dataset

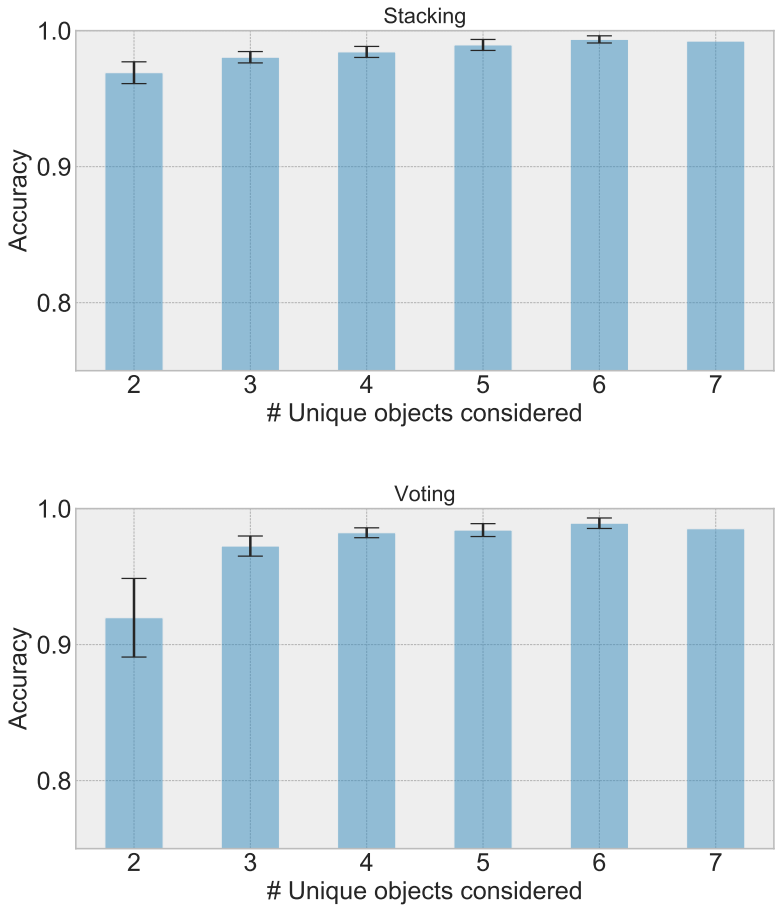


Figure 5.5: Averaged accuracy scores of different ensembles of unique objects in $H6$ for the voting meta-classifier and the COMBINED configuration. Each such ensemble is trained and tested separately, then the scores are averaged across the ensembles of the same type (*e.g.*, pairs, triples of unique objects). The 95% confidence intervals are given for each ensemble. The y-axis cutoff range is between 0.75 and 1.00.

is split using 10-fold cross-validation. Each test fold is used to evaluate a group of zero-effort attacks since it contains the samples of attackers' regular interactions with objects. The remaining attack samples are supplied to the zero-effort attack-trained classifier. To compare and evaluate the effectiveness of different types of attacks on the environment, I reported False Reject Rates (FRRs) at different thresholds of False Acceptance Rates (FARs). The FAR metric allows us to determine how many attempts Adv was successful in. On the other hand, FRR specifies how many legitimate samples from a victim have been misclassified as an attack. Note, that rather than completely preventing the user from executing a transaction this merely means that the user will

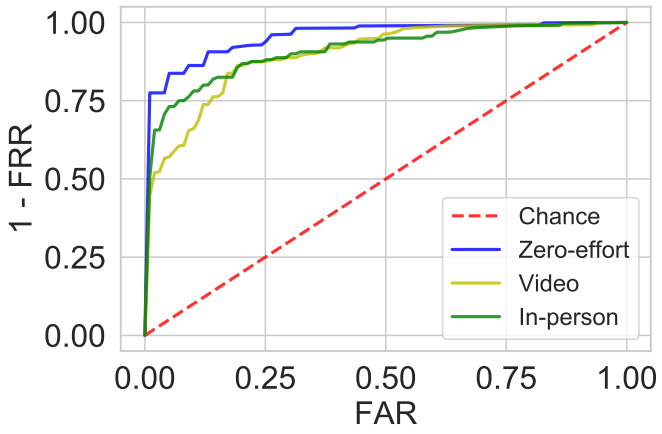
Table 5.8: False Reject Rates (FRRs) for interactions with different types of objects in respect to three kinds of attacks given different FAR thresholds. The ON-OBJECT column presents FRRs for the model with features extracted only from on-device sensors. OFF-OBJECT shows FRRs considering only features from co-located sensors, whereas COMBINED reveals FRRs for the model that uses the combined features from the co-located and on-device sensors. The results are averaged across all smart objects in my experiment.

		FAR = 10%	FAR = 1%
ON-OBJECT FRR	Zero-effort	0.1375	0.2486
	Video	0.3708	0.5812
	In-person	0.2250	0.5087
OFF-OBJECT FRR	Zero-effort	0.0063	0.0250
	Video	0.1750	0.1938
	In-person	0.1250	0.1500
COMBINED FRR	Zero-effort	0.0125	0.0250
	Video	0.1938	0.2500
	In-person	0.1500	0.2875

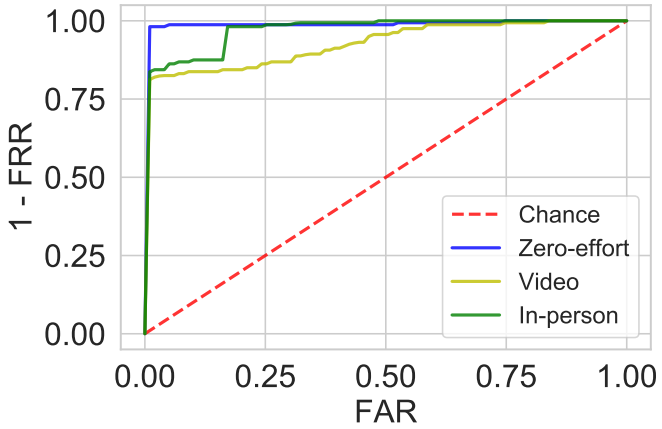
have to approve the transaction explicitly through their phone.

First, I examined FRRs for individual smart objects that the user interacts with. Next, I inspected the performance of ensembles of base classifiers that are responsible for interpreting different interactions with objects. Finally, I compared the performance of voting and stacking meta-classifiers by examining the receiver operating characteristic (ROC) curve for an ensemble of all available object interactions.

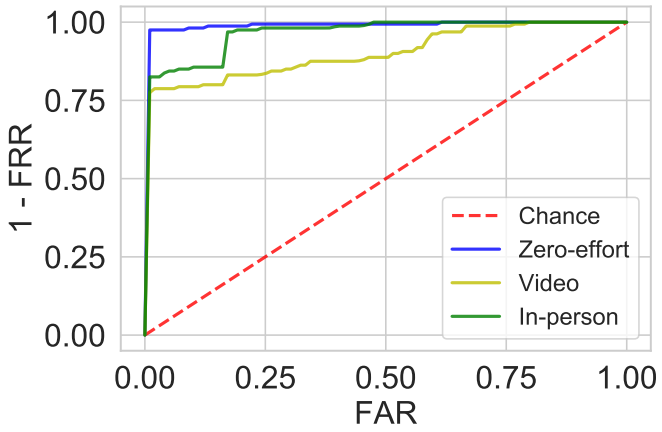
In Table 5.8, I present FRRs at 1% and 10% FAR thresholds averaged across all objects for three types of attacks targeting a dedicated user. Figure 5.6 shows their averaged ROC curves. Table 5.10 presents FRRs for individual smart objects in respect to zero-effort attacks without a dedicated victim, *i.e.*, the results are averaged across all users being considered a victim. For each attack, I calculated FRRs and FARs using different system configurations, including ON-OBJECT, OFF-OBJECT, and COMBINED. For OFF-OBJECT, only the top-performing features are selected. For each smart object, Table 5.9 shows what other objects these features were extracted from.



(a) ON-OBJECT



(b) OFF-OBJECT



(c) COMBINED

Figure 5.6: The plots above show the ROC curves for three system configurations respectively based on average FARs from single interaction types. Each curve represents a different group of attacks, *i.e.*, zero-effort (blue), in-person (green), and video-based (yellow) attacks.

Table 5.9: For the OFF-OBJECT configuration, I list all the co-located objects whose sensors provided the best features for individual object classifiers.

Object	Features from
Right cupboard door	Left cupboard door
Wide cupboard	Left cupboard door, Coffee machine, Microwave door
Left cupboard door	Right cupboard door, Wide cupboard, Oven door, Coffee machine, Pull-out drawer
Narrow cabinet	Right cupboard door, Wide cupboard, Left cupboard door, Oven door, Coffee machine, Pull-out drawer, Microwave door
Oven door	Right cupboard door, Wide cupboard, Left cupboard door, Narrow cabinet, Coffee machine, Pull-out drawer, Microwave door
Coffee machine	Right cupboard door, Wide cupboard, Pull-out drawer, Microwave door
Pull-out drawer	Right cupboard door, Wide cupboard, Left cupboard door, Oven door
Microwave door	Right cupboard door, Wide cupboard, Narrow cabinet, Coffee machine

In the training phase, I only used samples collected during participants' regular interactions with the smart environment. This is because I consider Adv who has access to the facilities. For example, a malicious co-worker whose typical interaction samples would be known by the system. A zero-effort attack, in which Adv does not attempt to mimic the behavior of a legitimate user, is an indication of the baseline performance of the system. Other types of attacks involve adversaries who either watched the video of the victim interacting with objects or observed the victim personally.

I observed that for authentication using OFF-OBJECT sensors, the system achieved an average false reject rate of less than 3% with a 1% false acceptance rate for zero-effort attacks. FRRs increase to 19% for video-based attacks and to 15% for in-person observation-based attacks, considering the same false acceptance rate. This means that even when defending against strong video-based attacks, the system does not require the user to explicitly approve transactions in more than 80% of cases, as the system can instead authenticate the user through their interactions with the smart environment.

Table 5.10: FRRs at two distinctive FAR thresholds for interactions with different types of objects in respect to zero-effort attacks given ON-OBJECT and OFF-OBJECT configurations. These configurations are compared to emphasize the improvement offered by considering co-located sensors. Presented results are averaged across all users being considered a victim.

Object Type	$FAR = 10\%$		$FAR = 1\%$	
	ON-OBJECT FRR	OFF-OBJECT FRR	ON-OBJECT FRR	OFF-OBJECT FRR
Right cupboard door	0.0526	0.0039	0.1401	0.0154
Wide cupboard	0.0577	0.000	0.2231	0.0077
Left cupboard door	0.0369	0.0039	0.1077	0.0039
Narrow cabinet	0.1141	0.0	0.2577	0.0039
Oven door	0.0305	0.0	0.1020	0.0
Coffee machine	0.0154	0.0	0.0731	0.0
Pull-out drawer	0.0385	0.0	0.1180	0.0
Microwave door	0.0987	0.0115	0.2962	0.0192

For the FAR of 10%, the FRR for zero-effort attacks drops to less than 1%. Similarly, FRRs for video-based and in-person attacks decrease to 18% and 13% respectively. The ON-OBJECT configuration exhibits the worst performance among all of the configuration types, resulting in false reject rates of 25% for the zero-effort attacks, 58%, and 50% for the other types of attacks. The COMBINED configuration guarantees better performance than ON-OBJECT, however, it exhibits worse performance than OFF-OBJECT due to the inclusion of features extracted from on-device sensors. It is noteworthy that the microwave door and the narrow cabinet classifiers perform significantly worse than others, which impacts the average scores. Since this effect is universal across users, this suggests that poorly-performing objects should be excluded by the meta-classifier.

Table 5.10 compares the performance of ON-OBJECT and OFF-OBJECT configurations across all smart objects. The narrow cabinet and the microwave door exhibit the worst FRRs in the ON-OBJECT configuration, resulting in false reject rates of 26% and 30% given a 1% false acceptance rate for zero-effort attacks. The FRRs drop to 0.4% and 1% when the model includes features extracted from co-located sensors. Since the OFF-OBJECT configuration exhibits the best performance, I focused on it for the remainder of this section.

The attackers from the video group could watch the video of the victim performing interactions with objects as often as desired for 24 hours. On the other hand, the attackers who observed the victim in person could follow them closely and look at the exact

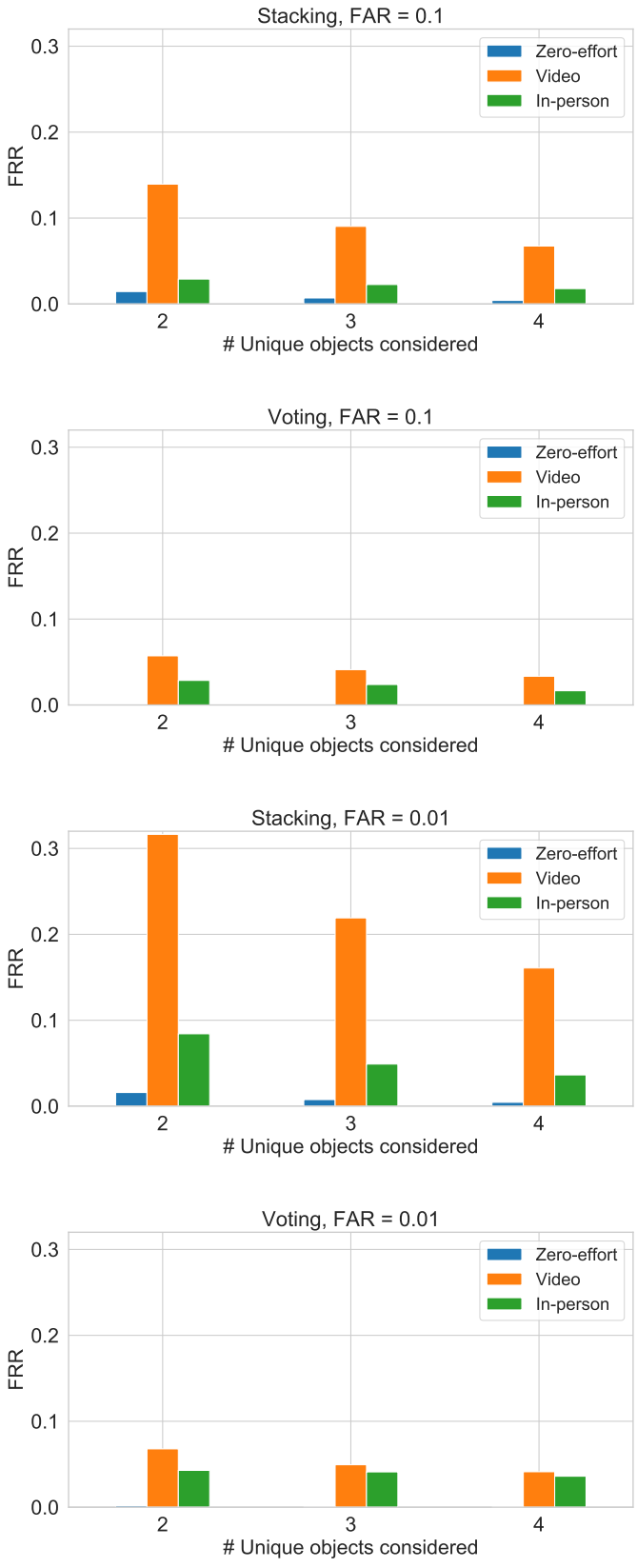


Figure 5.7: Averaged False Reject Rates (FRRs) at different False Acceptance Rates (FARs) thresholds calculated based on the performance of different ensembles of unique objects for two meta-classifiers and the OFF-OBJECT configuration. Each such ensemble is trained and tested separately, then the scores are averaged across the ensembles of the same type (*e.g.*, pairs, triples of unique objects).

body and hand movements. To understand this phenomenon, I asked the participants to describe their strategies. The participants from the video-based attack group watched the video three times on average before attempting to mimic the victim. When viewing the video, participants report that they paid attention to the strength with which the victim interacted with the objects, the use of the hands (left or right), the speed of the interaction, and the body position. The participants in the second group, on the other hand, focused mainly on the pace, strength, and rhythm of the interaction. All attackers focused their strategy on mimicking the power and speed with which the victim interacted with objects. Additionally, most of them attempted to spend a similar amount of time per interaction as the victim did. One of the attackers even counted the seconds spent on interactions with each object.

Considering multiple interactions with various objects can further improve the system's performance. Figure 5.7 shows averaged FRRs at two FAR thresholds of 10% and 1% for different ensembles of objects for the voting and stacking meta-classifiers given the OFF-OBJECT configuration. I focused on the OFF-OBJECT configuration here, as it exhibits the best performance out of the three considered configurations, and thus best demonstrates the potential performance gains that can be achieved. This could be further improved by adjusting the weights, *i.e.*, assigning smaller ones to interactions that exhibit worse performance. Generally, allowing the system to consider more interactions before authenticating the user results in better performance.

Overall, the voting method outperforms the stacking meta-classifier in this scenario. This method is also computationally less complex since it does not involve training another classifier with the predictions of the base classifiers. The voting meta-classifier achieves a false reject rate of less than 1% with a FAR of 1% whereas the stacking classifier obtains an FRR of 2% for the zero-effort attacks. The video-based attacks for the stacking classifier achieve an FRR of 32% when considering the ensemble of two unique objects given a FAR of 1%. On the other hand, the voting classifier obtains 8% FRR given the same FAR threshold. This means that for the voting classifier, the system can spare the user an explicit phone-based authentication in 92% of cases. I included

Table 5.11: Aggregated maximum values of RMI in percentages for the ON-OBJECT configuration, given different types of on-device sensors.

Object Type	ACC	MAG	GYRO	MIC
Right cupboard door	31.60	40.58	72.32	20.72
Wide cupboard	50.81	47.30	64.45	25.37
Left cupboard door	62.90	39.50	73.90	8.09
Narrow cabinet	30.39	33.28	19.41	15.09
Oven door	32.24	64.46	49.20	16.30
Coffee machine	34.07	41.64	60.48	41.67
Pull-out drawer	35.19	27.21	19.28	44.63
Microwave door	35.93	54.04	41.12	15.10

Table 5.12: Aggregated maximum values of RMI in percentages for the OFF-OBJECT configuration, given different types of co-located sensors.

Object Type	ACC	MAG	GYRO	MIC
Right cupboard door	78.63	71.49	72.08	17.33
Wide cupboard	85.86	53.72	59.10	28.07
Left cupboard door	77.69	75.53	74.21	17.49
Narrow cabinet	76.23	73.43	61.28	23.31
Oven door	74.67	73.52	60.76	17.93
Coffee machine	98.19	74.01	90.41	42.96
Pull-out drawer	80.61	69.31	56.81	25.50
Microwave door	86.09	54.78	58.37	20.33

only four smart objects in this analysis but considering more unique smart objects results in further improvements in the system performance.

5.6.3 Distinctiveness of sensor features

In order to judge the distinctiveness of features by different types of sensors, I used relative mutual information (RMI). RMI is defined as

$$\mathrm{RMI}(user, F) = \frac{H(user) - H(user|F)}{H(user)}$$

where H(A) is the entropy of A and H(A|B) denotes the entropy of A conditioned on B. Here, $user$ denotes the ground truth of the user performing the object interaction, whereas F is the vector of extracted features.

Tables 5.11 and 5.12 show the RMI for individual sensors that have been placed on objects as part of Experiment 2. These scores represent aggregated maximum values of RMI for a particular sensor on a specific object O_i, given different configurations

of the system. Each of these objects introduces a different way for a user to interact with the smart environment. Analysis of the distinctiveness of the features extracted from these sensors allows us to understand which ones contribute to better classification performance for a specific type of interaction. Each device has been equipped with an accelerometer (ACC), a magnetometer (MAG), a gyroscope (GYRO), and a microphone (MIC). Generally, I observed that the features extracted from GYRO and ACC exhibit high distinctiveness for most of the interaction types. For ON-OBJECT, the most distinctive features originate from GYRO whereas, for OFF-OBJECT, ACC appears to supply the most distinctive features. I also observed that, in many cases, the inputs from co-located objects generate higher RMI scores. On the other hand, the features extracted from MIC appear to have relatively low distinctiveness in comparison to other attributes for the majority of interactions.

Despite its generally low distinctiveness for most interactions, MIC achieves higher RMI values for interactions with the pull-out drawer and is the second most distinctive sensor for the coffee machine when I consider features extracted only from its on-device sensors. This can be explained as the drawer's contents making sounds continuously, changing based on how far extended the drawer is, whereas for most other events the main sounds were caused by the closing of doors—with little difference between users. Pressing the buttons of the coffee machine on the other hand makes faint sounds that differ between users with regard to the timing of the button presses.

GYRO shows particularly high distinctiveness for most interactions for ON-OBJECT, with the exceptions of the narrow cabinet and the pull-out drawer. The cabinet used in the experiment has a very stiff door that leads to abrupt openings with little variation between users. While this reduces the effectiveness of the recognition of users by sensors directly placed on the cabinet, such abrupt openings allow co-located sensors to capture stronger vibrations, hence, providing more accurate distinction. The lower RMI values for GYRO for the pull-out drawer can be explained by a lack of rotational movement. Instead, the most distinctive movement characteristics are the sounds and the acceleration which is why MIC and ACC are the most distinctive sensor types for this interaction.

ACC appears to provide the most distinctive features captured by co-located sensors. Interestingly, the vibration signals picked up by the co-located sensors exhibit the highest feature distinctiveness during interactions with the coffee maker. Overall, I noticed that OFF-OBJECT features provide better distinctiveness than the features gathered only by ON-OBJECT sensors. This suggests that the system can accurately authenticate users by their interactions with objects that do not have sensors directly placed on them.

5.7 Discussion

In my experiments, I limit interactions with any device to a single user at a time. This was necessary to obtain accurate identity labels to establish the distinctiveness of device interactions. This limitation, however, may lead to two potential problems in practice. If two users are interacting with different devices in the same room simultaneously or a short sequence, this may lead to decisions made using multiple device interactions being wrong. It can be avoided by only using interactions with the target device (the device requiring authentication) to make the decision. In addition, simultaneous interactions may affect the sensor signatures and make it harder to match fingerprints for either of them. However, simultaneous interactions are easily detected and either accounted for or ignored entirely.

Furthermore, due to time considerations and the unique requirements of the ongoing Covid-19 pandemic, I could only capture device interactions in a single session. This limits my analysis for different levels of FAR and FRR, as the total number of samples and attacker/victim pairs are too low to make a statistically robust analysis of extremely low FAR levels.

Another potential issue with the experiments is that sessions for different users were conducted one after the other. In theory, it would be possible for environmental effects to be present during one user's session but not for others, thereby leading to classifiers learning these effects as a proxy for user identity. For example, a sound pressure sensor may pick up increased ambient noise during a user's session. However, the fairly strong increase in FAR caused by imitation attacks (video and in-person) suggests that the classifiers capture (somewhat imitable) true user behavior as it is unlikely users would attempt to match the original environmental conditions during their

attack. Despite the promising results, there is room for improvement in this system and experimentation (see Chapter 7).

Overall, the analysis of the BeeHIVE system shaped the direction of the next research work. One of the key observations is the effect of device location on its ability to receive physical signals generated by the same user interaction. In the same vein, any event occurring in a smart environment can be observed simultaneously by many devices. It also includes events generated by other smart devices. Because it is difficult to predict the consequences and mutual influence of the elements of the smart ecosystem on each other, the interactions between these components should be monitored to prevent cross-channel attacks. Therefore, the next chapter focuses on modeling such device interactions in IoT environments to prevent potential misuse.

6

Butterfly Effect: Finding Cross-Channel Interactions in Smart Ecosystems using Dynamic Analysis

Contents

6.1 Motivation 108
6.1.1 Challenges 109
6.2 Related Work 110
6.2.1 Dynamic Analysis of Device Interactions 110
6.3 System Design 111
6.3.1 System Model 111
6.3.2 Adversary Model 112
6.4 Experimental Design 114
6.4.1 Testbed Overview 114
6.4.2 Data Collection 115
6.4.3 Experiment in an Unoccupied Environment 115
6.4.4 Experiment in an Occupied Environment 116
6.5 Methods 116
6.5.1 Causal Analysis 117
6.5.2 Knowledge Graphs 119
6.5.3 Ontology 119
6.5.4 Pre-processing 120
6.5.5 Graph Construction 121
6.5.6 Identifying Cross-channel Interaction Chains 121
6.5.7 Classification Metrics 122
6.6 Evaluation 122
6.6.1 Causal Analysis 123
6.6.2 Cross-channel Interaction Chains 132

6.7 Discussion 136

6.1 Motivation

Not only can users interact with smart devices, but these devices can also communicate with each other due to their heterogeneous nature. The previous chapter showed that co-located devices register the same user interactions through different channels. Moreover, based on the analysis of the capabilities of these devices in Chapter 3, I observed how many potential communication variants exist between them.

`Cross-channel` interactions occur between different communication channels of smart devices and indicate the effect of one device on another. Typical smart devices are equipped with various sensors; therefore they have many channels through which they can exchange information. For example, a Smart TV can be equipped with a motion sensor that can record the movement of other devices, including a robot vacuum cleaner. In this example, the movement of the vacuum cleaner is recorded by the motion sensor of the smart TV. If the smart TV has been configured to perform a certain action when motion is detected, it will be triggered by that interaction. While many of these interactions may be intended and expected by users, others may go unnoticed. Moreover, in some cases, such interactions can be abused to trigger unwanted chains of interactions between smart devices.

Existing solutions focus on static analysis and extracting information about communication channels from device descriptions or companion applications through natural language processing. However, since the quality and completeness of these descriptions cannot be guaranteed, and providing access to the source code of the companion application may not be desired by vendors, I propose a system that uses low-level physical signals extracted from the IoT hub to find potential interactions between channels. The main challenge in using static analysis to extract information about cross-channel interactions is the high false positive rate because even though two devices can potentially communicate over two channels, they may be spaced apart, so their performance may not actually be influencing each other. Hence, I introduce the following assumptions for this system:

O1. I assume that I have no access to the source code of companion applications.

O2. I do not trust the veracity or completeness of the device description.

O3. I must consider the physical limitations of device sensors.

Moreover, with this system, I want to be able to identify all of the cross-channel interactions that will lead to triggering automations on target devices. This, in turn, can facilitate the process of recommending rules and automations to the user to avoid potentially unsafe cross-channel interactions. I also want to be able to examine the impact of adding new automations in such dynamic environments.

6.1.1 Challenges

There are several challenges to consider while building a system that can meet the aforementioned objectives. First of all, to discover dependencies and potential violations in IoT environments, the system must be able to observe the real cross-channel interactions between different smart devices. Many existing systems extract information about communication channels from device descriptions or companion apps using methods such as NLP. Unfortunately, such descriptions are often incomplete. Consequently, I cannot assume that I will get a complete and correct overview of the cross-channel interactions between devices in such an environment. Moreover, integrating information from multiple companion applications can cause similar issues. Therefore, my system uses only the signals that the IoT hub receives from the smart devices and then analyses the mutual impact on the operation of these devices.

Because IoT ecosystems are dynamic, they require the use of methods and data structures that will facilitate analyzing the current state of such an environment efficiently. Since my system is platform agnostic and uses only physical signals obtained from the IoT hub for the analysis, it has to employ an algorithm that can accurately capture the dependencies between such signals—or, more precisely, find the causal relationships. During preliminary research, classical approaches proved ineffective because of the trend and seasonality of signals in IoT environments. Fortunately, there are alternative methods

for analyzing such signals. In this work, I decided to use multivariate transfer entropy (mTE) because it allows finding causal relationships for non-stationary time-series data.

Another challenge is the impact of user presence on the result of the causal analysis. As mentioned, some existing systems require users to leave the environment while training, and other systems assume their presence will not create an incorrect or incomplete image of cross-channel dynamics between the IoT devices. Since I consider such assumptions unrealistic, I present and discuss the actual impact of user presence during the training phase. I aim to find a trade-off between seamless user experience and training time while making my scenario as realistic as possible. I design this system such that, even in the presence of users, it will be able to create an actual representation of the IoT ecosystem.

6.2 Related Work

6.2.1 Dynamic Analysis of Device Interactions

Most of the research in this area focuses on using static analysis to model interactions between smart devices. These solutions are based on extracting information about the channels of such devices from their description or companion applications. A more recent work, IoTGuard [146], proposes to use dynamic analysis to confirm and verify the results of static analysis. The aforementioned system adds interaction monitoring logic to the mobile application's source code during the static analysis phase and then blocks actions that could lead to potential safety violations. This solution is not optimal as modifying the source code results in a huge development overhead. Moreover, the use of static analysis produces a lot of inaccuracies, so the system detects many false interactions. This, in turn, stops IoTGuard from allowing benign actions to occur in the smart environment.

Another work, IoTSafe [147], proposes modifying the companion app's source code to provide contextual data regarding the location of the smart devices in the environment. In this approach, the system executes every automation in isolation and analyses its impact on other devices. The cross-channel interactions are pre-determined during the static analysis stage through the use of natural language processing. Since each interaction is examined separately, dynamic analysis of the smart environment is time-consuming, *i.e.*

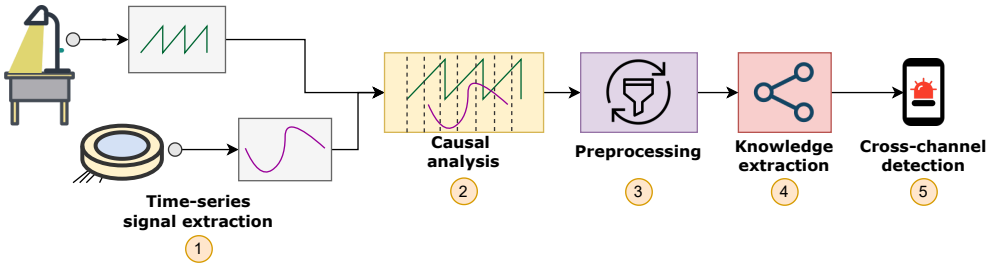


Figure 6.1: An overview of the system. First, the system extracts time-series signals from smart devices (1). These signals are then analysed to find triggers—*i.e.*, relevant sources R, for a target T (2). Based on the outcome of this analysis and after the pre-processing of the data (3), the knowledge graph is built (4), and the system flags potential cross-channel violations to users (5).

taking 40 minutes to verify 5 cross-channel interactions. Such a solution does not scale well, especially considering larger environments. In addition, as in the solution discussed earlier, extracting channels through a static analysis not only results in false positives but may also omit channels that were not included in the description (intentionally by malicious developers or as an accidental omission). Overall, the static analysis does not consider the consequences of physical placements of smart devices; thus, fails to include signal propagation properties. This leads to rejecting rules and policies that otherwise are safe. The metadata is supplied by the users, which is prone to user error; hence, omitting crucial chained interactions.

Due to the limitations of static analysis and the aforementioned systems, I propose a system based solely on dynamic analysis. Moreover, my approach is the only currently existing solution that can operate with the humans present in the smart environment.

6.3 System Design

6.3.1 System Model

In this thesis, I consider an IoT ecosystem where smart devices are equipped with various sensors that enable interactions between channels. Such interactions are usually used to build automations that facilitate the daily activities of users. For example, when a user enters a room and the motion sensor detects their movement, the lights turn on. Such ecosystems are dynamic, which means that new devices can be added and existing devices can be relocated at any time. When setting up new automations, users are often not fully

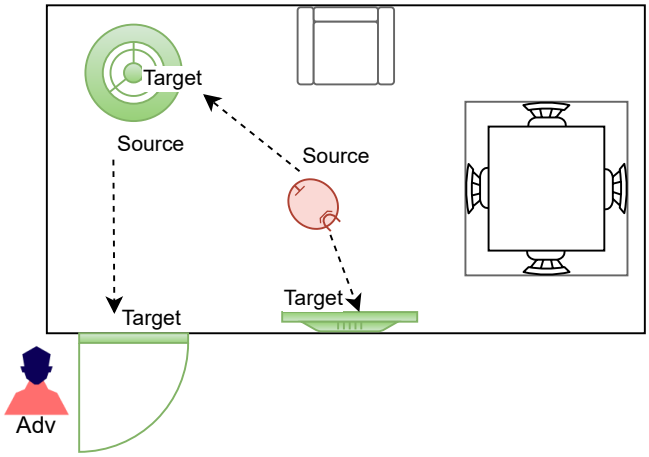


Figure 6.2: The diagram presents my threat model. The adversary (Adv) compromises a vulnerable device (in red) to target other devices (in green). This leads to triggering automation that opens the window, allowing Adv to enter the house.

aware of the impact these automations may have on other nearby smart devices. This is for a variety of reasons, including the fact that devices from different vendors are often controlled with separate companion applications, creating a decentralized structure that is difficult to supervise. The only element connecting individual smart devices is an IoT hub that aggregates their data. The lack of appropriate tools to monitor these ecosystems and the inability to control the flow of information between different communication channels of devices opens up the possibility to abuse such infrastructure.

In order to address the existing challenges and achieve my stated goals, I propose a system consisting of four main components. Figure 6.1 shows these components. The first one extracts time-series signals generated by individual devices from an IoT hub (1). The second component performs causal analysis to find existing relationships between device channels (2). As a result, the system identifies all source and target pairs where the source affects the target signal. Next, it pre-processes the data (3) and then constructs a knowledge graph using the ontology described in Section 6.5 (4). Finally, it calculates and displays the paths to all smart devices to help in identifying safety-critical violations (5).

6.3.2 Adversary Model

The cross-channel interactions between smart devices can be used by an adversary (Adv) to trigger unwanted events. Figure 6.2 shows an example of such an event. The Robot

Vacuum Cleaner (in red) has a known vulnerability that allows Adv to compromise it and turn it on. While moving, the Robot Vacuum Cleaner generates motion that can be picked up by other devices (in green), including Motion Sensors, that are set to turn on the TV or smart lights. Finally, the window is configured to unlock if the lights are on and someone walks into the room. Since the Motion Sensors cannot distinguish between movement generated by humans and smart devices, the movement of the Robot Vacuum Cleaner will activate this automation.

In my threat model, Adv's main objective is to trigger safety-critical automation by abusing existing cross-channel interactions in the IoT ecosystem. Although Adv does not have physical access to the IoT infrastructure, nor can they compromise the companion apps or an IoT hub, Adv can learn what devices are present in such an environment and attempt to exploit their vulnerabilities. Moreover, the adversary can trick the user into installing a malicious companion app to recommend and set new automations that could cause violations. I assume that the smart devices that store and process sensitive data, such as smart cameras or smart locks, have better security controls than other types of smart devices, and thus are out of reach for Adv. Furthermore, I assume that the system remains uncompromised during the initial setup and training process.

To summarize, the assumptions of the adversary model are as follows:

- Adv cannot physically access the IoT infrastructure.
- Adv cannot compromise the companion apps or IoT hub.
- Adv can learn about the devices present in the environment and their vulnerabilities.
- Adv can install a malicious companion app.
- Adv cannot interrupt the training phase of the system.

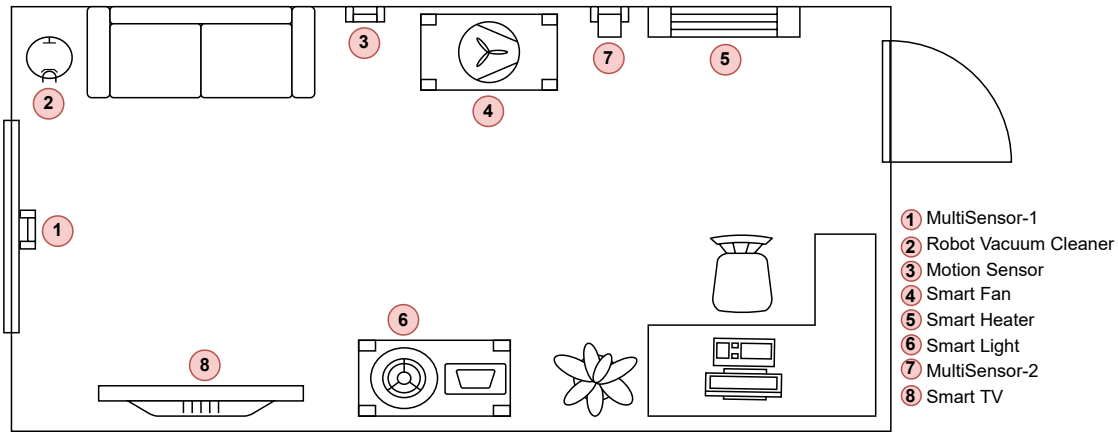


Figure 6.3: This diagram provides a representation of the room in which I conducted the experiments, indicating device placement.

6.4 Experimental Design

6.4.1 Testbed Overview

For my system to be usable to smart environment users, I had to consider the following factors:

- The accuracy in detecting cross-channel interactions between the devices,
- The time constraints,
- The user's active presence in the smart environment.

In order to assess the performance of the proposed system, a single room was transformed into a small-size smart environment. This environment consisted of 8 devices that are commonly found in smart homes and offices. Two different experiments were performed to analyze and compare the impact of the user's active presence in the environment. In the first experiment, the smart environment was *unoccupied*; hence, no user was present. In the second one, the environment was *occupied* with the user carrying out their regular activities inside it. Figure 6.3 shows the physical configuration and positions of smart devices in the room. These smart devices and their physical channels are listed in Table 6.1. The device channels were extracted from the device descriptions provided by the manufacturers. I decided to include them in this table only to demonstrate the extent of

my experiment. I would like to note that I do not use this in the data analysis phase. A more detailed description of the methods used in this work is given in Section 6.5.

6.4.2 Data Collection

The data was gathered using the Samsung SmartThings cloud platform, which is often used instead of the physical smart hub. This platform monitors smart device data in real-time. Each device consists of different capabilities that are broken down into attributes and commands. Attributes relate to the states of the device, while commands refer to functions that the device can execute. To be able to centrally control smart devices, I connected the official SmartThings companion app to the smart hub. This constituted the interface between the devices and the user and allowed the collection of a stream of timestamped device data.

6.4.3 Experiment in an Unoccupied Environment

For this experiment, 10 automations were configured. Each automation consists of a trigger and action. The trigger may be represented by a specific change in a physical channel (*e.g.*, the temperature raises above a certain threshold). Some changes to device channels, such as motion, can propagate quickly. On the other hand, some changes on other channels, *e.g.*, temperature or humidity, may propagate slower, with a delay. To evaluate the performance of this system and assess how well it detects causal relationships between physical channels of smart devices, a wide range of device channels were included in the automation set. These automations are shown in Table 6.2. To create a more realistic environment, some devices and their channel automations have not been configured. Overall, the data collection took 102 minutes in this experiment. This data has been divided into time intervals of 15, 30, 45, 60, and 90 minutes, with each segment starting at the same timestamp. The observations have been segmented this way to create five independent interaction models to compare the accuracy of the causal analysis with regard to the amount of time taken to perform this analysis.

6.4.4 Experiment in an Occupied Environment

In this experiment, I wanted to examine the impact of the user's presence in the smart environment on the dynamic analysis of cross-channel interactions. More specifically, I wanted to determine how similar and accurate the observations made by the system would be when the user interacts with smart devices during the analysis.

This experiment took place in the same environment (but on a different day) where the aforementioned experiment was conducted. Moreover, I used the same set of automations as before. This time, the data collection took 260 minutes. This experiment was extended because I assumed that the system may need more time to detect cross-channel interactions when the user is present. This user had to perform the following actions:

1. Turn off ④ Smart Fan manually within the first 5 minutes,
2. Interact with smart devices naturally and comfortably (inc. watching ⑧ Smart TV or working),
3. Leave the room for 5 and 10 minutes every two hours.

These instructions ensured that the user was actively engaging and interacting with smart devices to simulate a real human user in a smart home. Notably, the first instruction was included to impact the flow of automations, as compared to the unoccupied experiment, in which the fan triggered a cascade of events. The collected data was divided into time intervals of 30, 60, 120, and 240 minutes.

6.5 Methods

My system performs a dynamic analysis of signal propagation in IoT ecosystems, which helps detect potential cross-channel interactions. After extracting a set of time-series signals S from the IoT hub, the system examines the relationships between these signals to determine causality. Then, based on this analysis, a knowledge graph is built with which the system finds triggers for various automations set by users. Moreover, this data structure is used to identify the chains of events leading to violations in IoT ecosystems and to help prevent potential cross-channel attacks.

Table 6.1: The devices used in my experiments and the physical channels that they sense.

Device Index	Name	Physical Channels	Capabilities
①	MultiSensor-1	Humidity Luminosity Motion Temperature UV	`temperature` (value) `humidity` (value) `illuminance` (value) `powerSource` (value) `uvIndex` (value) `battery` (value) `motion` (on, off) `tampered` (tampered, notTampered)
②	Robot Vacuum Cleaner	-	`switch` (on, off) `powerConsumption` (value) `operatingState` (value) `remainingTime` (value) `robotCleanerMovement` (value)
③	Motion Sensor	Motion Temperature	`battery` (value) `temperature` (value) `motion` (on, off)
④	Smart Fan	-	`powerMeter` (value) `energyConsumption` (value) `switch` (on, off)
⑤	Smart Heater	-	`powerMeter` (value) `energyConsumption` (value) `switch` (on, off)
⑥	Smart Light	-	`colourTemperature` (value) `colourControl` (value) `dimmer` (value)
⑦	MultiSensor-2	Temperature Vibration	`temperature` (value) `vibration` (value) `3-axissensor` (value) `battery` (value)
⑧	Smart TV	-	`switch` (on, off)

6.5.1 Causal Analysis

Initially, I used Granger's formula [148] to find causal relationships between time-series signals generated by smart devices. However, this approach produced inconclusive results because the time-series data were not stationary, so I had to search for alternative methods. Hence, to analyze the cross-channel dynamics between smart devices, I chose multivariate transfer entropy (mTE) [149]. This technique models causal relationships between non-stationary time-series signals, such as those generated by smart devices. I chose this method because other algorithms do not consider the multi-dimensionality and directionality of time-series signals, which is crucial in detecting which device

Table 6.2: The automations used in my experiments, with reference to devices listed in Table 6.1. The last two trigger at the starting time and repeat on a 20-minute counter.

Automation Index	Automation
A1	if (3) motion.on, then turn on (8)
A2	if (3) motion.on, then send (2) back to its charger
A3	if (3) motion.off, then turn off (8)
A4	if (1) motion.on, then turn on (6)
A5	if (1) motion.off, then turn off (6)
A6	if (7) temperature $>=$ 85°F, then turn on (4)
A7	if (7) temperature $>=$ 85°F, then turn off (5)
A8	if (7) temperature $<$ 83°F, then turn off (4)
A9	if time $=$ 20 and (3) motion.off, then turn on (2)
A10	if time $=$ 20 and (3) motion.off, then turn on (5)

properties will affect other devices in an IoT ecosystem. Due to the use of mTE, my system is able to recognise the synergy between many signal sources $S = \{S_i, i = 1, \ldots, n\}$. This means that my system can analyze how physical signals generated by smart devices affect the channels of a single target device. I denote an individual time-series signal on the target device's channels as t. The set of relevant sources R (*i.e.*, signals that impact the target device's channels) is iteratively built by testing the source signal's past variables $p \in P$ bounded by the number of lags and maximizing the conditional mutual information $I(p; t_m|R)$ where t_m is the target signal's current value. Moreover, this algorithm determines the optimal number of observations, a lag l_x, that helps capture the causal relationships between the signals. In this work, I used the Python IDTxl implementation of this algorithm, which employs a greedy iterative method for estimating causality [150]. This algorithm reduces redundancy and takes into account interactions between different signals.

6.5.2 Knowledge Graphs

A knowledge graph is a data structure that represents entities (*i.e.*, nodes) and their relationships (*i.e.*, edges) [151]. To construct such a graph, I used the NetworkX Python package [152]. The knowledge graph is particularly helpful in integrating and representing knowledge from multiple sources. It also enables one to derive knowledge based on implicit or partial information. For example, it helps to infer the relationship between nodes that are not directly connected. This is especially helpful in analyzing dynamic ecosystems such as IoT environments where certain chains of events can lead to actions being taken on a device that does not share a direct channel with other devices. In this work, I build and use the knowledge graph to learn about physical interactions between smart devices and, at a more detailed level, between their physical channels. The knowledge graph can also accommodate the dynamic, constantly evolving character of smart environments. During the initial construction of such a graph, not all smart devices may be available or present in a given environment. Users can add new devices and set up new automations at any time. Therefore, the data structure must be able to quickly and easily adapt to all changes.

6.5.3 Ontology

A crucial element enabling the quick implementation of such changes is the knowledge graph ontology. The ontology defines the basic structure of a knowledge graph—*i.e.*, its schema. It generalizes graph instances and introduces classes that can be used not only to visualize but also to cluster different types of nodes. Figure 6.4 shows the ontology I created. This schema maps smart devices and their capabilities to graph instances. This enables the creation of an abstract representation of the causal dynamics in the IoT ecosystem. Such a graph can be used to find direct and indirect relationships between various device nodes. In this work, the system constructs a knowledge graph with two types of nodes. The first type represents a smart device (*e.g.*, a motion sensor), while the second type represents channels, denoted as capabilities, that the device can modify (which includes physical channels, such as motion, and functions that monitor internal states, such as battery level). Edges between two channel nodes embed information

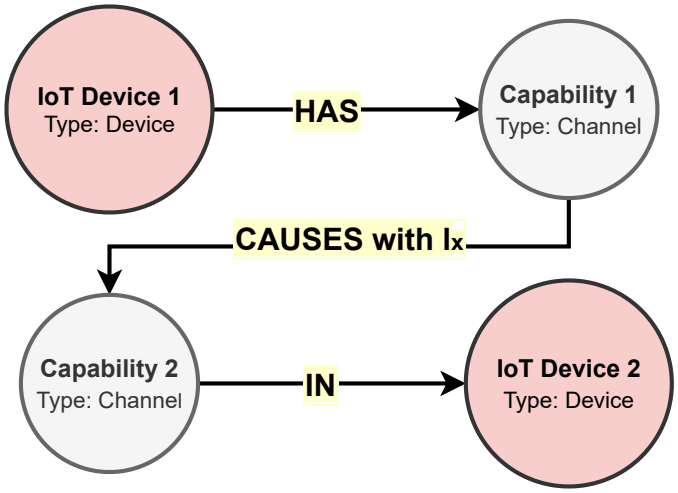


Figure 6.4: The diagram provides a general overview of the schema of the knowledge graph. Two types of triple exist in this schema. The first one consists of a device and capability pair along with an edge representing a directional relationship between the two nodes. The second one involves the capability-to-capability relationship with a lag (l_x) added as a graph edge parameter.

about the lag l_x. The resulting graph is a directed graph that shows how and in which direction the signal propagates through the IoT ecosystem.

6.5.4 Pre-processing

Before constructing a knowledge graph to find unique paths to smart devices in the IoT environment, my system performs filtering of the connections between channels of the same type. I assume that these channels cannot trigger one another. For instance, although the system may interpret that an increase in temperature on one device causes an increase in temperature on another smart device, this would be an incorrect cross-channel interaction as two sensors can only capture events but do not generate any. Further information about types of device channels can be found through the API.

The system removes the channels that contain constant values—*i.e.*, the values that do not change over time. It also filters out the connections between channels of the same device. In this work, I am interested in capturing which smart devices and channels can be used to trigger others. Some devices will be configured to respond to certain events and their channels will likely impact one another. For example, a Multipurpose Sensor can be set to turn on a Smart Heater when motion is detected. Since this device has

a temperature sensor, it would create a connection between its motion sensor and the temperature sensor. Thus, my system removes such graph connections as erroneous.

6.5.5 Graph Construction

After determining R for each target T, the system decomposes this set into triples (S_i, T, l_x). These triples are then incorporated into a knowledge graph. Since there are two types of nodes (device and sensor nodes) at each iteration, three triples are created. The first triple connects the source device node D_S with the source channel node S_i. The second triple represents a directed relationship between S_i and the target channel node T. Lastly, T connects to the target device node D_T. Since the first and last triple represents non-causal relationships, l_x is empty. With such a graph structure, the system can promptly compute all paths between D_S and D_T, or S_i and T on a highly granular level. This helps to reveal non-direct connections between channels and devices. For instance, if motion detected by a Motion Sensor switches on a Smart Light and a Smart Fan triggers the Motion Sensor, the system can find a simple path in the knowledge graph that starts with the Smart Fan and leads to the Smart Light.

6.5.6 Identifying Cross-channel Interaction Chains

To find or predict cross-channel interaction chains (*i.e.*, paths) to a selected T, the system uses the depth-first search algorithm [152]. A set of paths to a given target F_T are computed based on the outcome of the mTE analysis and the current state of the knowledge graph. When the user adds new automation between existing channels of two smart devices, the new link connecting two channel nodes is created and the new path in the graph is added to F_T. This way, the system can capture unwanted interactions in the IoT ecosystem by informing users about the potential consequences of adding certain automation. This is then refined by re-running the mTE analysis to observe the actual impact of the new automation. A set of unique paths for each smart device is defined as an ordered list of device nodes that can be traversed to reach another device node.

6.5.7 Classification Metrics

To measure the performance of my system, I classify the interactions identified by the system into three types. A *true positive* (TP) is an interaction that the system has identified correctly—*i.e.*, where the interaction describes an automation in Table 6.2 or physical impact that one device has on another. A *time-coupled false positive* (FP-TC) is an interaction that has been identified incorrectly, strictly as a result of co-occurring automations—*i.e.*, where two or more automations trigger at the same time, every time, and the system cannot fully resolve the causality. I subdivide the false positives because FP-TCs arise from the environment setup and can therefore be mitigated either by knowledge of automation timings (which could be incorporated into the knowledge graph construction) or by decorrelating events by adding a small random epsilon to automation timings. In my experiment, I set `A9` and `A10` to repeat at regular intervals to ensure that some FP-TCs are caused in order to measure the impact. The other type of commonly occurring false positives is a *message-reordering false positive* (FP-MR). An FP-MR is an interaction that has been identified incorrectly due to a mismatch between the event timestamp information used by the app, which is responsible for the automations, and the timestamps given by the cloud API, which is used for data logging. As both of these have separate clocks and use the receipt of a message to determine its timestamp, messages may arrive in a different order at the cloud platform than in the app. As Goksel *et al.* [153] points out, these kinds of misordered events commonly occur in hybrid smart home systems and can be security threats in and of themselves. However, for systems such as mine, this issue could be mitigated by having one central arbiter that is responsible for event timestamp information.

6.6 Evaluation

In this section, I discuss the performance of my system. The first part focuses on the results of the causal analysis in an unoccupied IoT environment. In the second part, I compare these results with those of the same analysis performed in an occupied ecosystem. The

Table 6.3: The training times in minutes for different training durations. The duration indicates the amount of time the data was collected during the experiment.

Duration (mins)	15	30	45	60	120	240
Unoccupied	16	53	86	105	–	–
Occupied	–	54	–	75	151	511

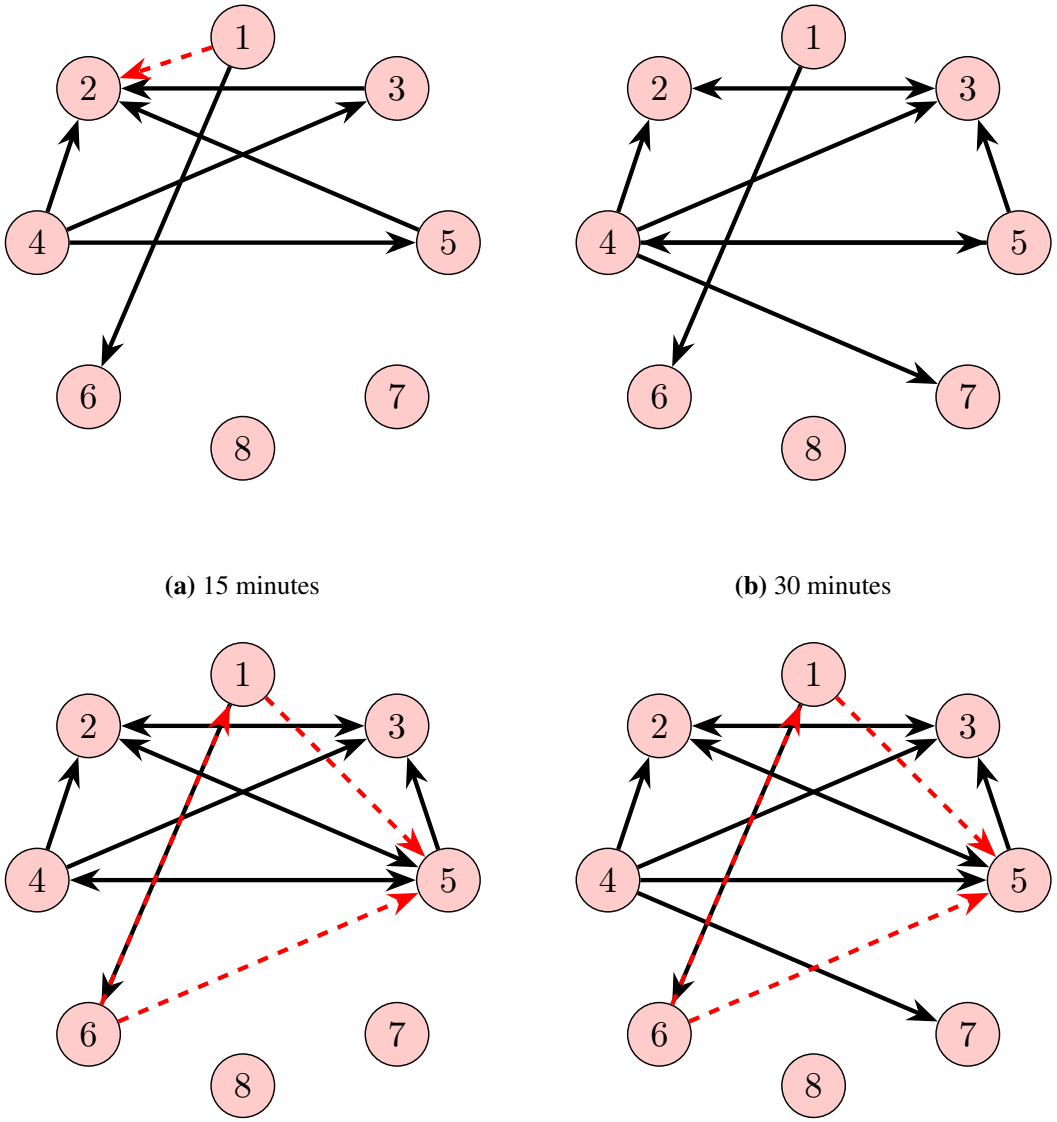
included knowledge graphs show direct connections between the device nodes without their capability nodes to aid the comparison.

I show how long it takes for the system to generate knowledge graphs and train hyperparameters for samples of different durations. Finally, I show and evaluate the cross-channel interactions detected by the system. The experimental setup as well as the placement of smart devices in my IoT environment is described in Section 6.4.

6.6.1 Causal Analysis

The system performed a causal analysis on collected samples of different duration. My main goal was to observe how much time it took for the system to create a graph that shows the cross-channel interactions between smart devices recorded over a certain period (*e.g.*, 15 minutes). In other words, I wanted to see how long it takes for various signals to propagate. During this process, the system tuned its hyperparameters by testing a different number of lags to capture the relationships between the time-series signals. Figure 6.5 shows a comparison between the knowledge graphs created for the unoccupied and occupied IoT environments in time intervals of 15, 30, 45, and 60 minutes for the unoccupied environment and 30, 60, 180, and 240 minutes for the occupied environment. Table 6.3 outlines how long it took the system to create these knowledge graphs and select hyperparameters for each time interval. Interestingly, it takes almost twice as long to analyze a sample of a given length. This means that the duration of the observation should be kept relatively short.

I also note that because the `A9` and `A10` automations switch on (2) Robot Vacuum Cleaner and (5) Smart Heater simultaneously, I expected this to be reflected in the causal analysis as a time-coupled false positive.



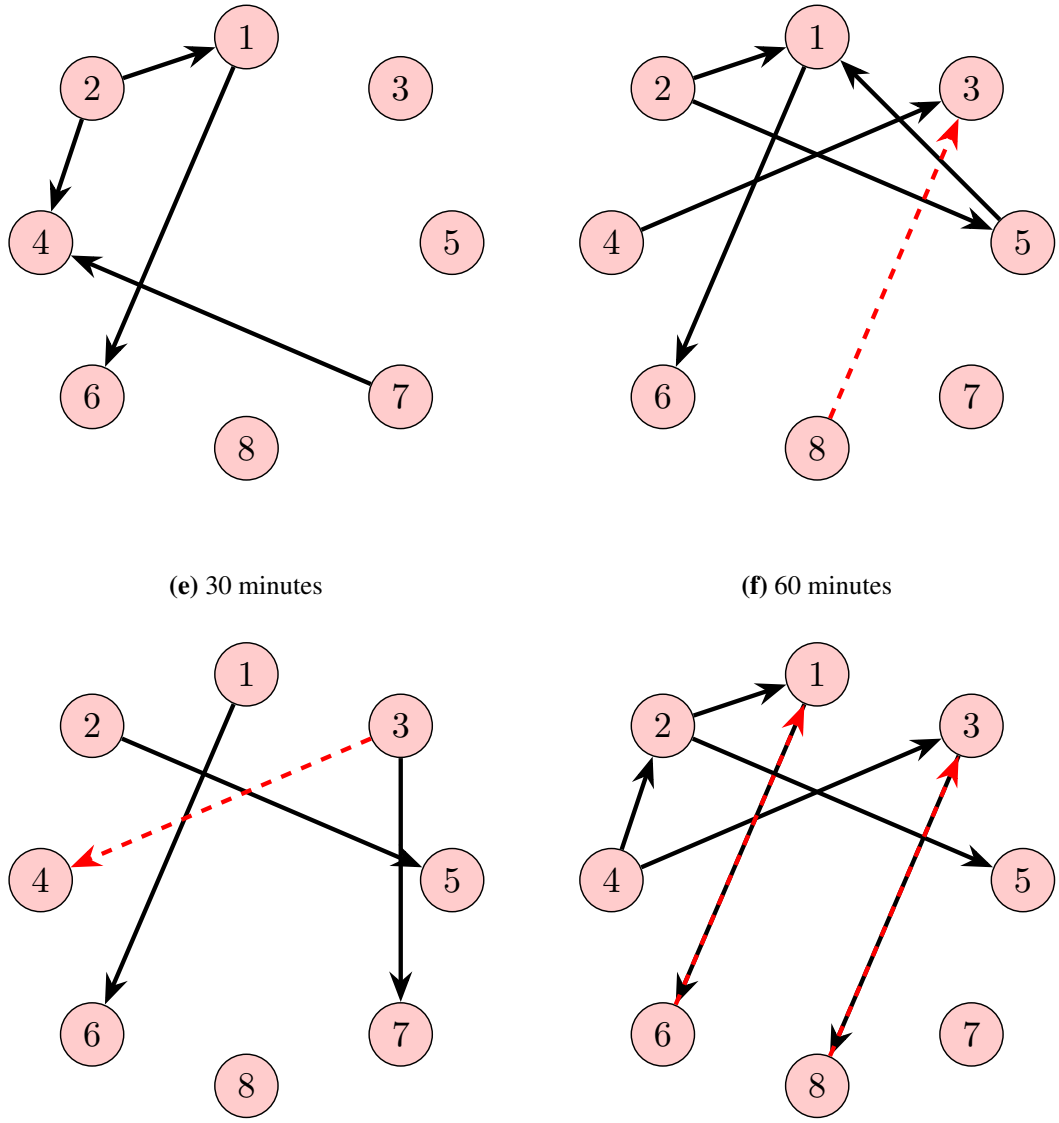
(a) 15 minutes **(b)** 30 minutes

(c) 45 minutes **(d)** 60 minutes

(a)–(d) unoccupied environment

Figure 6.5: Comparison of knowledge graphs built based on the causal analysis of cross-channel interactions between smart devices during observations of different duration. False positives are depicted with red dashed arrows. The first part shows the graphs formed in the unoccupied IoT environment (a–d) in four time intervals of 15, 30, 45, and 60 minutes. *(cont.)*

Unoccupied Environment. The first row in Figure 6.5 shows the knowledge graphs built as a result of the causal analysis between the time-series signals collected by the IoT hub in the unoccupied environment. My first observation concerns the impact of the length of data collection on the resulting graphs. Depending on the duration of the



(e) 30 minutes **(f)** 60 minutes

(g) 120 minutes **(h)** 240 minutes

(e)–(h) occupied environment

Figure 6.5: *(cont.)* The second part presents the graphs constructed in the occupied environment (e–h) in four time intervals of 30, 60, 120, and 240 minutes.

intervals, the outcome of the causal analysis varies between the graphs. During the first 15 minutes of observation, due to the physical arrangement of smart devices in the room and thus the propagation limitations of their signals, not all interactions appear on the graph. For example, graph (a) does not show how the motion of ② Robot Vacuum Cleaner affected ③ Motion Sensor. This is because ② Robot Vacuum Cleaner did

not reach ③ Motion Sensor in the first 15 minutes. It was, however, captured by the system over a 30-minute time period (b) in which ② Robot Vacuum Cleaner began to vacuum the area next to the door.

Cause	Trigger	Smart Device	Triggered	Smart Device
A6	Motion	④ Smart Fan	→	③ Motion Sensor
A2 | A6	Temperature, Motion	⑤ Smart Heater	→	② Robot Vacuum Cleaner
A2	Motion	③ Motion Sensor	→	② Robot Vacuum Cleaner
A4 | A5	Motion	① MultiSensor-1	→	⑥ Smart Lights
A7	Motion	④ Smart Fan	→	⑤ Smart Heater
A8 | A9	Motion	④ Smart Fan	→	② Robot Vacuum Cleaner
–	FP	① MultiSensor-1	→	② Robot Vacuum Cleaner

Table 6.4: The cross-channel interactions identified by the system after 15 minutes of observation in the unoccupied environment.

The summary of the findings obtained from graph (a) can be found in Table 6.4. The system correctly registered cross-channel interactions between smart devices ①, ②, ③, and ⑥, which resulted from three automations, namely A2, A4 and A5. Interestingly, A2 was triggered by the movement of ④ Smart Fan. This is an example of an unintended interaction that my system found. It detected that ④ Smart Fan triggered ③ Motion Sensor and, as a result of the A9 automation, it also affected the operation of ② Robot Vacuum Cleaner. The interaction between ④ Smart Fan and ⑤ Smart Heater was captured by the system because of A7. Here, the state of ⑤ depends on whether ③ Motion Sensor observes movement; hence, in some cases, the oscillations of ④ Smart Fan can impact the operation of ⑤ Smart Heater. In addition, the system correctly inferred that ⑤ Smart Heater was triggering ② Robot Vacuum Cleaner because of A2 and A6. If the temperature in the room increases, ④ turns on and sends ② back to the charging station. On the other hand, the system incorrectly identified that ① MultiSensor-1 triggers ② due to a range of automations (*i.e.*, A2, A6, A7, A9, and A10) causing the temperature channel of ① to register changes in temperature when ② was cleaning.

Table 6.5 displays the cross-channel interactions that have been extracted by the system after 30 minutes of observation. It registered the same set of automations: A2, A4 and A5. Similarly to (a), graph (b) shows that the operation of ④ Smart Fan affected ③ Motion Sensor, ② Robot Vacuum Cleaner, and ⑤ Smart Heater. However, this time

Cause	Trigger	Smart Device	Triggered	Smart Device
A6	Motion	④ Smart Fan	→	③ Motion Sensor
A2	Motion	③ Motion Sensor	→	② Robot Vacuum Cleaner
A4|A5	Motion	① MultiSensor-1	→	⑥ Smart Lights
A6|A8	Temperature, Motion	⑤ Smart Heater	→	③ Motion Sensor
A7	Motion, Temperature	④ Smart Fan	→	⑤ Smart Heater
A6|A8	Temperature	⑤ Smart Heater	→	④ Smart Fan
A8|A9	Motion	④ Smart Fan	→	② Robot Vacuum Cleaner
–	Motion	② Robot Vacuum Cleaner	→	③ Motion Sensor
A6|A7	Motion	④ Smart Fan	→	⑦ MultiSensor-2

Table 6.5: The cross-channel interactions identified by the system after 30 minutes of observation in the unoccupied environment.

the system also correctly identified that ② Robot Vacuum Cleaner triggers ③ Motion Sensor when it is vacuuming the room. The graph (b) also shows that ⑤ Smart Heater triggers ③ Motion Sensor because of the movement of ④ Smart Fan that ③ registers due to A6 and A8. My system identified the relationship between ④ Smart Fan and ⑦ MultiSensor-2. This is because ⑤ Smart Heater turns off when ④ turns on as a result of A6 and A7. The interaction between ⑤ Smart Heater and ④ Smart Fan was captured by the system because of A6 and A8. This is because when the temperature in the room increases, ④ turns on. In comparison to the previous graph, it also detected that ⑤ Smart Heater interacts with ④ Smart Fan. This is the result of the A6 and A8 automations that turn on ④ Smart Fan when the temperature exceeds a certain threshold.

Cause	Trigger	Smart Device	Triggered	Smart Device
A6	Motion	④ Smart Fan	→	③ Motion Sensor
A2	Motion	③ Motion Sensor	→	② Robot Vacuum Cleaner
A4|A5	Motion	① MultiSensor-1	→	⑥ Smart Lights
A6|A8	Temperature, Motion	⑤ Smart Heater	→	③ Motion Sensor
A7	Motion, Temperature	④ Smart Fan	→	⑤ Smart Heater
A6|A8	Temperature	⑤ Smart Heater	→	④ Smart Fan
A8|A9	Motion	④ Smart Fan	→	② Robot Vacuum Cleaner
–	Motion	② Robot Vacuum Cleaner	→	③ Motion Sensor
A10	Temperature	⑤ Smart Heater	→	② Robot Vacuum Cleaner
A10	Motion	② Robot Vacuum Cleaner	→	⑤ Smart Heater
–	FP	① MultiSensor-1	→	⑤ Smart Heater
–	FP	⑥ Smart Lights	→	⑤ Smart Heater
–	FP	⑥ Smart Lights	→	① MultiSensor-1

Table 6.6: The cross-channel interactions identified by the system after 45 minutes of observation in the unoccupied environment.

Table 6.6 summarizes the cross-channel interactions detected after 45 minutes of observation, as depicted in graph (c). In addition to the interactions mentioned before, the

system identified the interaction between ② Robot Vacuum Cleaner and ⑤ Smart Heater due to A10. Moreover, it found that ⑥ Smart Light affects the operation of ⑤ Smart Heater and ① MultiSensor-1. These two interactions, however, are message-reordering false positives, which can be explained by the limited reporting capabilities of smart devices as well as messages arriving in a different order at the companion app and the IoT hub. My system incorrectly identified that ① MultiSensor-1 triggered ⑤ Smart Heater due to associating the movement of ② with the increase of the temperature caused by the operation of ⑤. This is classified as an FP-TC.

Cause	Trigger	Smart Device	Triggered	Smart Device
A6	Motion	④ Smart Fan	→	③ Motion Sensor
A2	Motion	③ Motion Sensor	→	② Robot Vacuum Cleaner
A4|A5	Motion	① MultiSensor-1	→	⑥ Smart Lights
A6|A8	Temperature, Motion	⑤ Smart Heater	→	③ Motion Sensor
A7	Motion, Temperature	④ Smart Fan	→	⑤ Smart Heater
A8|A9	Motion	④ Smart Fan	→	② Robot Vacuum Cleaner
–	Motion	② Robot Vacuum Cleaner	→	③ Motion Sensor
A10	Temperature	⑤ Smart Heater	→	② Robot Vacuum Cleaner
A10	Motion	② Robot Vacuum Cleaner	→	⑤ Smart Heater
A6|A7	Motion	④ Smart Fan	→	⑦ MultiSensor-2
–	FP	① MultiSensor-1	→	⑤ Smart Heater
–	FP	⑥ Smart Lights	→	⑤ Smart Heater
–	FP	⑥ Smart Lights	→	① MultiSensor-1

Table 6.7: The paths identified by the system after 60 minutes of observation in the unoccupied environment.

The final graph (d), which is summarized in Table 6.7, shows a new connection between ④ Smart Fan and ⑦ MultiSensor-2. I saw this connection before in graph (b), but due to cross-channel interaction over the temperature channel. However, this time it was caused because ④ generated vibrations while operating, which were picked up by the accelerometer of ⑦. After 60 minutes of observations, I can also see two incorrectly identified interactions between ⑥ Smart Light, ① MultiSensor-1, and ⑤ Smart Heater. I also notice a false positive between ① and ⑤ that resulted from A9 and A10. Despite that, my system was able to capture 10 correct interactions.

Overall, by comparing the graphs, I notice that the longer observation periods expose different cross-channel interactions between the smart devices, but also introduce false positives.

Occupied Environment. In the second part of the experiment, the system analyzed the same IoT environment with the same set of automations, but with a user present in the room. The user was performing their regular activities, so some events were not triggered by the automations or were stopped by the user. My goal was to analyze the difference in generated knowledge graphs and compare the number of cross-channel interactions captured by the system.

Cause	Trigger	Smart Device	Triggered	Smart Device
–	Motion	② Robot Vacuum Cleaner	→	① MultiSensor-1
A6|A10	Motion	② Robot Vacuum Cleaner	→	④ Smart Fan
A6|A8	Motion	⑦ MultiSensor-2	→	④ Smart Fan
A4|A5	Motion	① MultiSensor-1	→	⑥ Smart Lights

Table 6.8: The cross-channel interactions identified by the system after 30 minutes of observation in the occupied environment.

Table 6.8 summarizes cross-channel interactions detected by the system after the initial 30 minutes of observation. The system correctly detected that ① MultiSensor-1 impacts the operation of ⑥ Smart Light, which results from A4 and A5. In addition, the interaction between ② Robot Vacuum Cleaner and ① MultiSensor-1 was identified. Interestingly, this interaction in an unoccupied environment was not observed at all. This system also correctly identified that the movement of ② Robot Vacuum Cleaner impacts ④ Smart Fan due to A6 and A10. Another correctly identified cross-channel interaction is between ⑦ MultiSensor-2 and ④ Smart Fan as a result of A6 and A8. Again, this is the first time this interaction appears. During this run, the system did not capture many interactions related to ④ Smart Fan. This was caused by the user manually turning ④ off and on while passing by ③ during this time interval.

When comparing (e) to (b), I notice that the system detected significantly fewer cross-channel interactions, due to the user's actions in the environment. This prompted us to introduce larger observation intervals for the occupied environment to allow the system to experience the patterns in physical signals created by the automations.

After 60 minutes of observation in the occupied environment, the system finally detected the interaction between ④ Smart Fan and ③ Motion Sensor, which is shown in (f) and Table 6.9. However, due to the movement of the user, the system mistakenly

Cause	Trigger	Smart Device	Triggered	Smart Device
–	Motion	② Robot Vacuum Cleaner	→	① MultiSensor-1
A10	Motion	② Robot Vacuum Cleaner	→	⑤ Smart Heater
A6	Motion	④ Smart Fan	→	③ Motion Sensor
A4|A5	Motion	① MultiSensor-1	→	⑥ Smart Lights
–	Temperature	⑤ Smart Heater	→	① MultiSensor-1
–	FP	⑧ Smart TV	→	③ Motion Sensor

Table 6.9: The cross-channel interactions identified by the system after 60 minutes of observation in the occupied environment.

identified that ⑧ Smart TV also affects ③. When a user enters the room, both ③ and ① log this motion, so A1 triggers. Interestingly, the system removed the connection between ④ Smart Fan and ⑦ MultiSensor-2, so the A6 and A8 automations were not detected. Compared to (d), the system missed many cross-channel interactions, especially with ② Robot Vacuum Cleaner. This is mainly because the user's movement in the environment triggered ③ quite often, which prevented ② from operating. Therefore, the environment needed more time to register the interactions with ② than in the unoccupied environment. Moreover, the system struggled with capturing interactions with ⑤ Smart Heater. Again, this was caused by the user's interventions (*e.g.*, opening the door when the temperature in the room increased significantly). Similarly to (c) and (d), graph (f) shows that the system incorrectly detected an interaction between ⑥ Smart Light and ① MultiSensor-1. Overall, the system detected 6 more cross-channel interactions in an unoccupied environment over the same 60-minute period.

Cause	Trigger	Smart Device	Triggered	Smart Device
A10	Motion	② Robot Vacuum Cleaner	→	⑤ Smart Heater
A10	Motion	③ Motion Sensor	→	⑦ MultiSensor-2
A4|A5	Motion	① MultiSensor-1	→	⑥ Smart Lights
–	FP	③ Motion Sensor	→	④ Smart Fan

Table 6.10: The cross-channel interactions identified by the system after 120 minutes of observation in the occupied environment.

After two hours of observation, the system detected only three valid cross-channel interactions and a false positive, which can be seen in the graph (h) and Table 6.10. Due to the A4 and A5 automations, the system captured interactions between ① MultiSensor-1 and ⑥ Smart Light. The system also correctly identified that ③ Motion Sensor impacts

⑦ as a result of A10. In addition, it registered that ② Robot Vacuum Cleaner impacts ⑤ Smart Heater because of A10. In comparison to other graphs, it did not capture how ④ Smart Fan affects ③. Instead, it registered the opposite, which is an FP-MR.

Cause	Trigger	Smart Device	Triggered	Smart Device
A10	Motion	② Robot Vacuum Cleaner	→	⑤ Smart Heater
–	Motion	④ Smart Fan	→	③ Motion Sensor
–	Motion	② Robot Vacuum Cleaner	→	① MultiSensor-1
A4|A5	Motion	① MultiSensor-1	→	⑥ Smart Lights
A1|A3	Motion	③ Motion Sensor	→	⑧ Smart TV
A9	Motion	④ Smart Fan	→	② Robot Vacuum Cleaner
–	FP	⑥ Smart Lights	→	① MultiSensor-1
–	FP	⑧ Smart TV	→	③ Motion Sensor

Table 6.11: The cross-channel interactions identified by the system after 240 minutes of observation in the occupied environment.

However, after observing the occupied environment for four hours, the system was able to identify six correct interactions between the IoT devices and only 2 false positives in total, which is summarized in Table 6.11. For the first time in the occupied environment, the system detected the dependency between ④ Smart Fan and ② caused by A9. In comparison to (g), the system detected three more cross-channel interactions and one more false positive from the FP-MR category.

Overall, I note that in the occupied environment the system had to observe the ecosystem for longer to filter the connections created by user interventions. I have summarised the results in Table 6.12. The best performance is observed at the time interval of 30 minutes for the unoccupied environment, where the system detected 10 cross-channel interactions and no false positives. On the other hand, the system identified four fewer cross-channel interactions between the smart devices in the 4-hour interval in the occupied environment. Interestingly, due to human interventions, there are no false positives from the FP-TC category, which originate from configuring two automations to turn on smart devices simultaneously, in the occupied environment. Thus, in a typical IoT environment, which is a hybrid of unoccupied and occupied environments, the performance of my system could inherently improve further, leading to faster detection of invalid cross-channel interactions.

Table 6.12: The number of correctly identified cross-channel interactions (TPs) and the number of incorrect interactions from two categories of false positives (FP-TC and FP-MR) in an unoccupied environment at intervals of 15, 30, 45, and 60 minutes, and in an occupied environment at intervals of 30, 60, 120, and 240 minutes.

	Unoccupied			**Occupied**		
	TP	FP-TC	FP-MR	TP	FP-TC	FP-MR
15 mins	6	0	1	-	-	-
30 mins	10	0	0	4	0	0
45 mins	10	1	2	-	-	-
60 mins	10	1	2	5	0	1
120 mins	-	-	-	3	0	1
240 mins	-	-	-	6	0	2

In summary, the system needs more time to detect cross-channel interactions in the occupied environment due to user interference. This is especially apparent for the channels that capture signals that take more time to propagate (*e.g.*, temperature). In my experiment, most false positives resulted from the simultaneous activation of ⑤ Smart Heater and ② Robot Vacuum Cleaner. From the perspective of causal analysis, such time signals could cause each other because their values started changing at a similar time. In the actual IoT environment, such connections would be refined after a longer observation of the ecosystem and due to the user's natural interaction with the operation of these devices.

6.6.2 Cross-channel Interaction Chains

Another function of my system is to find all unique cross-channel interaction chains (*i.e.*, paths) that can be used to trigger devices in an IoT environment using generated knowledge graphs. I define a unique path as an ordered list of nodes that can be traversed to access the node of interest. Tables 6.13 and 6.14 show the number of unique paths identified for each smart device in unoccupied and occupied environments at intervals of 60 and 240 minutes, respectively. I also discuss what specific device channels were affected for each device and I list all devices that can be used to trigger others.

Unoccupied Environment. The generated knowledge graphs show direct connections between the channels of various devices in the IoT ecosystem. However, listing only direct connections is not enough, as sometimes the operation of one device may trigger

Table 6.13: The number of unique paths identified to each smart device over a 60-minute interval in an unoccupied environment.

Device Index	Name	Unique Paths	Channels	Triggering Devices
①	MultiSensor-1	1	`motion`	⑥
②	Robot Vacuum Cleaner	15	`robotCleanerMovement` `operatingState` `remainingTime`	① ③ ④ ⑤ ⑥
③	Motion Sensor	17	`motion`	① ② ④ ⑤ ⑥
④	Smart Fan	-	-	-
⑤	Smart Heater	11	`energyConsumption` `switch`	① ② ③ ④ ⑥
⑥	Smart Light	1	`switch`	①
⑦	MultiSensor-2	1	`temperature`	④
⑧	Smart TV	-	-	-

a chain of events. In the unoccupied environment, the system detected 30 unique paths between smart devices. Among these paths, 21 paths are identified correctly and 9 paths are incorrect due to one FP-TC connection and two FP-MR connections in the graph (d). As mentioned, these false positives are either introduced by configuring automations to run two devices at the same time (*e.g.*, `A9` and `A10` in my environment) or by messages arriving in a different order at the smart hub and the companion app. The former can be solved by users naturally interacting with the devices around them, thus interrupting the simultaneous operation of these devices from time to time. However, I note that some smart devices may always run simultaneously, in which case the problem may persist. The latter can be dealt with by having a central authority for message timestamps between independent decision-making modules, such as companion apps and IoT hubs.

In the unoccupied environment, the system identified that ① MultiSensor-1 can be triggered by ⑥ Smart Light. This is a false positive. Furthermore, the system detected that there are 15 paths to trigger ② Robot Vacuum Cleaner through its 3 channels. The device can be impacted by the operation of ③ Motion Sensor and ④ Smart Fan due to `A2` and `A9`. Moreover, it can also be triggered by ⑤ Smart Heater because of `A6` and

A8. Among these unique paths, 8 of them that originate from nodes ⑥ and ① are false positives. There are 3 direct connections to ② from the aforementioned devices and 4 indirect ones from ④ or ⑤ through ③ to ② as well as from ④, ⑤, and ③ to ②.

The system also identified 17 unique paths to ③ Motion Sensor. Similar to the previous device, there are 8 false positives originating from ⑥ and ①. Apart from that, ③ can be triggered by ② Robot Vacuum Cleaner through its motion. Moreover, its operation can be impacted by the oscillation of ④ Smart Fan, which can also be triggered by ⑤ Smart Heater. This device can be triggered through 3 direct cross-channel interactions and 6 chains of interactions.

Next, the system detected 11 unique interaction chains affecting the operation of ⑤ Smart Heater. There are 4 false positives originating from ⑥ and ①. ⑤ can be triggered by ③ Motion Sensor, ② Robot Vacuum Cleaner, and ④ Smart Fan because of A10. This device has two direct connections in the graph but 5 chains of interactions may impact its operation in this IoT ecosystem.

The system also identified that there is one path to trigger ⑥ Smart Light and it originates from ① MultiSensor-1 due to A4 and A5. Moreover, there is one path detected for ⑦ MultiSensor-2. The temperature sensor of this device can be affected by ④ Smart Fan. In the unoccupied environment, the system was unable to detect any interactions with ⑧ Smart TV or ④ Smart Fan.

Occupied Environment. In the occupied environment, the system identified 13 unique paths, including 11 valid paths and 2 false positives. These two false positives are caused by the two incorrect connections between ⑥ Smart Light and ① MultiSensor-1 as well as the connection between ⑧ Smart TV and ③ Motion Sensor.

The system correctly identified that ① can be triggered by both the movement of ② Robot Vacuum Cleaner and ④ Smart Fan. Graph (h) does not have an edge between ① and ④, which means that ④ can affect it through its impact on ②. Thus, the system infers that ④ influences the operation of ① because of A2 and A9. This system also detected a third path originating from ⑥, however, this one is a false positive.

Next, the system identified only a single path to trigger ② Robot Vacuum Cleaner. I can observe that because of the motion of ④, the remainingTime channel of ②

Table 6.14: The number of unique paths identified to each smart device over a 240-minute interval in an occupied environment.

Device Index	Name	Unique Paths	Channels	Triggering Devices
①	MultiSensor-1	3	`motion`	② ④ ⑥
②	Robot Vacuum Cleaner	1	`remainingTime`	④
③	Motion Sensor	2	`motion`	④ ⑧
④	Smart Fan	-	-	-
⑤	Smart Heater	2	`powerMeter` `switch`	② ② ④
⑥	Smart Light	3	`switch`	① ② ④
⑦	MultiSensor-2	-	-	-
⑧	Smart TV	2	`switch`	③ ④

is impacted. Again, this is because of two automations that can switch ② on and off depending on the detected movement of ④.

According to the system, ③ Motion Sensor can be triggered by two devices. One of these cross-channel interactions is correct, the other is a false positive. ③ can be impacted by the movement of ④. However, because the connection between ⑧ Smart TV and ③ was incorrectly inferred, there is no real impact of ⑧ on ③.

Graph (h) shows no cross-channel interactions that trigger ④ Smart Fan; thus, there are no unique paths leading to it from other nodes. On the other hand, I can see that ⑤ Smart Heater can be influenced by the operation of ② Robot Vacuum Cleaner, as well as by ④. This is correct because of `A10`. Interestingly, rather than detecting the direct connection between ③ Motion Sensor and ⑤, the system flagged that ④ can impact ⑤ through its impact on the operation of ②. Presumably, prolonged observation of the occupied environment would allow the system to detect this direct interaction.

Furthermore, the system identified 3 unique paths to ⑥ Smart Light. ① MultiSensor-1 switches ⑥ on and off due to `A4` and `A5`. The other two cross-channel interactions do not appear as direct connections in the graph. This system detected that ② Robot Vacuum Cleaner can trigger ⑥ by generating movement that is detected by ① that turns ⑥ on. Similarly, because ④ can impact the motion of ②, it can be used to trigger a chain of events that will impact the operation of ⑥.

Finally, the system found two paths to trigger ⑧ Smart TV. The `A1` and `A3` automations cause ⑧ to be turned on when motion is detected by ③ Motion Sensor. And, since ④ Smart Fan can generate movement that is captured by ③, it can also do so.

In comparison to an unoccupied environment, the system did not detect any paths to ⑦ MultiSensor-2. However, it detected that two devices can trigger ③ Smart TV, which was not shown in the unoccupied environment. In the occupied environment, the system needs significantly more time to identify the unique paths to smart devices.

6.7 Discussion

Dynamic testing can take a considerable amount of time in which the user cannot use the space normally, or at the very least might be constrained in how they use the space. It is therefore desirable to keep this time as short as possible. This has been a severe limitation of related work, and it still partially applies to my system. However, in comparison to IoTSafe [147], which takes about 12.5 hours to complete its training process, the system can detect every interaction and automation after only 30 minutes of training time without user interference. It has to be noted, though, that the experimental setup of IoTSafe includes more devices and automations than the environment used in this work. In contrast to the related work, my system also allows for user presence during the training period, extending the training time to four hours to find the majority of interactions, but without forcing the user to abandon the IoT environment during this time. In future work, I could investigate how this can be used to even greater effect by enabling continuous learning of cross-channel interactions in user-occupied spaces (more on this in Chapter 7).

In addition, pure dynamic approaches can only detect interactions based on observed behavior. Therefore, rare interactions, or interactions with very specific preconditions,

might be missed by this system. This is a general problem that related work solves through the use of static analysis. This, however, leads to an over-approximation of potentially risky interactions, as described before. I argue that my proposed system is much more usable as it does not overload the user with warnings of hypothetical situations but rather flags up demonstrable interactions, thus providing a tangible security benefit.

Overall, after performing two experiments to evaluate the system's performance under various conditions, I have arrived at the conclusion that the system possesses the capability to accurately detect cross-channel interactions, even with the user actively engaged within the environment, following an extended observation period. This significant finding represents a groundbreaking development, as it marks the first-ever cross-channel detection system of its kind that can operate in the presence of an actively engaged user in a smart environment. Furthermore, the system has been designed with a specific emphasis on incorporating a knowledge graph, which enables it to readily adapt and scale in order to fulfill the demands of ever-changing and continuously expanding smart infrastructures. It is worth noting that while the system can detect certain types of automations instantly, there are some exceptions that require a longer observation period. For instance, channels like temperature or humidity may take longer to propagate, which means that the system cannot identify causal relationships immediately. Additionally, some automations may be configured to work in conjunction with one another, causing all these signals to be classified as sources for certain cross-channel interactions. In Chapter 7, I delve into further detail regarding the potential solutions to this problem.

This and the previous chapter discussed systems that focus on different types of interactions in IoT ecosystems. Through this work, I wanted to show how the heterogeneity of smart devices can be used to secure such environments, taking into account different attack vectors. Both systems were designed to provide a seamless user experience and not rely on users' active participation.

7 Conclusions

Contents

7.1 Summary 139
7.2 Impact 142
7.3 Future Work 144

7.1 Summary

Due to the variety of built-in sensors, smart devices possess many capabilities. As such, they can convert raw physical signals of different origins into digital data. For example, an air quality monitor can measure temperature, air pressure, and carbon dioxide build-up in a room. On the other hand, a smart power meter can monitor energy and power consumption, which can allow users to reduce energy costs in their homes. Smart devices, although made by different vendors, can be centrally integrated through a smart hub. Users can either directly control such devices in the smart ecosystem by interacting with them or remotely by using a companion application.

Due to the complexity of such environments, there are many attack vectors, ranging from individual devices to mobile applications, or communication with a smart hub. One of the key problems is the lack of adequate access control mechanisms. Moreover, due to the multi-channel nature of these devices and the lack of robust methods of controlling the

flow of information in smart environments, they are at risk of violations of cross-channel interactions. In this dissertation, I investigated how the heterogeneity of the Internet of Things devices can be used to protect smart environments from the aforementioned threats.

After a preliminary study, I realized that I need to collect realistic data in user environments in order to study access control systems. Therefore, I introduced a remote experimentation framework that enables the collection of data in the natural environment for the target group and tackles the basic challenges that arise when experimenting in such uncontrolled environments. I evaluated this framework using a case study, in which I applied this approach to a user study on smart object interactions for a biometric authentication and identification system introduced in this thesis. My experiment took place under lockdown conditions and I showed that by deploying this framework, I was able to reach a wider group of participants and effectively create a usable dataset without the need for direct interaction with the participants. In general, the key realization was that remote experimentation allows for flexible real-world data collection that involves a diverse set of participants with different needs and backgrounds, and a varied set of locations, each with its unique parameters, including size and layout. Since such experiments are no longer bound to one specific location maintained by the researchers, they could be scaled both quantitatively and qualitatively.

Smart devices understand common contexts despite their differences in capabilities. Consequently, they can, for example, detect events taking place in the environment, such as when the user is cooking or sleeping. I hypothesized that these devices would be able to recognize not only what users were doing but also their identity. Thus, I have introduced a system to authenticate users in smart environments based on naturally occurring interactions with objects around them. Notably, such a system does not require any sensors on the object itself but makes use of sensors placed arbitrarily in the room. Using the aforementioned remote experimentation methodology, I conducted two experiments. The first experiment, with 25 participants in 6 different households, showed that regardless of the size of the household or a specific household setup, users can be distinguished with high accuracy. The proposed system is able to identify users with an accuracy of at least 91% on average if just one interaction is being recorded by

the co-located sensors. This identification accuracy increases to 100% when multiple interactions are considered by the system.

I have conducted the second experiment in real-world settings with a total of 13 participants, which shows that using these kinds of smart object interactions for authentication is also feasible. This is a crucial finding because there is a need for stronger authorization and access controls in such environments, but many smart devices offer only limited interfaces to implement security features. Therefore, current systems often rely on cumbersome app-based authentication methods that require the user to always have their phone at hand. The proposed system can complement such phone-based authentication methods and reduce how often a user has to explicitly approve a transaction in the smart home companion app. I show that the system demonstrates good authentication performance against zero-effort attacks, with less than 1% of transactions requiring external approval at a FAR of 1% when considering a single object interaction.

When attackers attempting to imitate the victim's behavior after observing them in-person or through video footage are considered, the user has to approve more transactions explicitly to maintain a 1% FAR. However, the system can still authenticate more than 80% of transactions unobtrusively when considering video-based attackers, rising to 85% of transactions for in-person attacks. I also show that the system's confidence in the authentication decision can be significantly improved if more than one object interaction is considered. Including more interactions with objects can further increase the authentication success rates to 92% even when considering the strongest attacker. These promising results and the potential for easy deployment make this behavioral biometric system a good candidate to improve the security of smart environments seamlessly and unobtrusively. I made my entire dataset and the code needed to reproduce the results available online to allow other researchers to build on my work.

Due to the increase in the number of smart devices in homes and offices, as well as the automation of their operation to provide greater comfort to users, there is a growing need to properly secure such environments not only through user authentication, as automations also may be used by an adversary to exploit various devices. The users of the smart environment are often not fully aware of the impact devices have on each other

due to the chains of cross-channel interactions. While previous research has focused on static analysis to detect such interactions, I show that better results can be achieved with dynamic analysis without having to rely on device descriptions and natural language processing. Thus, in this dissertation, I introduce a system that performs causal analysis using physical signals extracted from device sensors to learn cross-channel interactions in IoT ecosystems. This system collects such signals from a smart hub and uses them to build a knowledge graph representing the ecosystem. The generated knowledge graph is analyzed to find chains of interactions that can affect other devices in the environment.

I conducted two experiments to evaluate the performance of this system. In the first experiment, the system observes an unoccupied environment with a set of pre-configured automations. In the second one, the user is present in that environment. The system identified 10 cross-channel interactions resulting in 30 unique chains of interactions in the first experiment. On the other hand, it detected 6 such interactions leading to 13 chains of interactions in the occupied environment. This shows that the system can detect such interactions after prolonged observation even with the user present in the environment. It is the first cross-channel detection system that can operate in the presence of the user who is actively engaging with the smart environment.

7.2 Impact

The solutions introduced in this thesis have the potential to enhance security in smart environments, but they also give rise to several privacy concerns. Firstly, the proposed systems collect and store sensitive behavioral data that could track the user's movements and reveal their personal information, such as daily routines and habits. Therefore, it is critical to ensure that this type of data is securely collected and stored to prevent unauthorized access.

Furthermore, the BeeHive system could be used for continuous authentication, which could lead to constant surveillance in homes or workplaces. Therefore, it is important to make sure that users are aware of the system's capabilities and that their privacy is respected by household members or employers. Moreover, integration of this system with other smart devices, such as smartphones and smartwatches, could allow it to collect even

more data. This integration raises concerns about cross-device tracking and the potential aggregation of data from multiple sources, which could further compromise user privacy.

Additionally, this thesis presents a novel methodology for conducting biometric studies remotely, which simplifies the process of setup and experimentation by utilizing a modular, reusable, and scalable framework. This framework provides researchers and practitioners with the ability to conduct experiments outside of traditional research facilities, reducing costs and increasing accessibility. Furthermore, the framework's modular design allows for easy integration of new components and modules, making it adaptable to various research needs and experimentation requirements. However, this framework could be expanded to serve particular research subfields and better protect the user's privacy.

While these concerns may have significant implications for users, they can be alleviated by implementing the proposed solutions in a privacy-preserving manner. This involves limiting the storage and use of data to its original purpose and ensuring that users are informed about the extent to which their personal information is shared with companion applications and vendors.

Despite the limitations of the proposed systems, they signify an advancement over existing solutions and encourage further dialogue on potential improvements. Overall, the BeeHive system holds significant implications for the field of behavioral biometrics and the progress of more secure and convenient authentication methods in smart environments. By leveraging object interactions, this system can gather behavioral data in a non-intrusive and passive manner, which is less susceptible to errors compared to traditional authentication methods. This data can be utilized to create a user profile and verify the user's identity based on their distinct behavior without requiring their active participation.

In general, the limitations and potential improvements of the presented solutions include resolving privacy concerns associated with the collection and storage of sensitive behavioral data, enhancing system accuracy, expanding performance analysis to incorporate unknown users, and investigating the impact of attackers on the training phase of the cross-channel detection system. For more information on future work, refer to the following section.

7.3 Future Work

During the study of the proposed systems and solutions presented in this thesis, I identified some extensions and improvements that could be made. In this section, I focus on discussing the potential for future work.

The framework for remote experimentation could be expanded in several directions. For example, one could further improve the authenticity of the collected data by not only transferring the experiment to a real-world location but also by weaving the data collection into every day activities by monitoring participants over time or with gamification techniques. Moreover, the framework could be further instantiated to serve particular research subfields such as behavioral biometrics, human-computer interaction, human activity recognition, and beyond. The instantiation could involve such experimental variables as the types of remote locations, equipment used, data collected, or analysis performed and would likely improve the reproducibility of the experiments even further.

Moreover, given the promising results shown by the analysis of the proposed biometric identification and authentication system, the user study could be extended to collect an additional large-scale dataset. This study would also include users with disabilities to investigate whether the system provides good performance for people with mobility impairments. In addition, this system currently does not account for behavioral changes over time. This means that a new experiment can be set up to examine variations in user interactions after a certain period. Presently, the system can separate between known smart environment users. Therefore, the performance analysis of this system could be extended to include unseen users.

The analysis of the cross-channel detection system can also be improved. The dynamic nature of smart environments means that systems such as this one must continuously adapt and periodically retrain as new devices are added, existing devices are removed or displaced, and other features of the environment are changed. My system has to rerun the training sequence to validate the impact of newly added automations. In my analysis, I did not consider an attacker that is able to interfere with the training process. Nonetheless, a powerful attacker might be capable of exploiting this process to poison the system in

a way that enables future attacks to remain unnoticed. In future work, I could examine how such an attacker might affect this cross-channel detection system, and how such systems can mitigate the impact of an attack on the training process.

Appendices

A

Instructions for User Study

Contents

A.1 BeeHIVE – Information 149
A.2 BeeHIVE – User Manual 150

A.1 BeeHIVE – Information

Before the commencement of the user study, participants had received the following information:

"The primary objective of this experiment is to determine whether the patterns in the data collected from commonly used sensors in smart environments can be used to seamlessly identify individuals when they operate smart devices and perform their daily activities. You will be asked to repeat the same sequence of interaction tasks with smart devices 20 times. An example of an interaction task includes opening and closing a fridge or dishwasher door (assuming that the sensor is placed on its door). In one round you will be asked to interact with every device once (e.g. open and close the kitchen door, walk to the dishwasher, open and close it, then go to the fridge, open and close its doors, etc.). Every household member will be asked to repeat the same sequence every time. Due to the enforcement of a national lockdown, the experiments will be

conducted in participants' private homes. The equipment (i.e. a mobile device and sensors) will be delivered to the address you provide, and you will be given instruction sheets explaining how to set up the experiment. The consent forms and information sheets will be sent to you by e-mail. The data will be stored locally on the devices as well as the server that belongs to our research group."

A.2 BeeHIVE – User Manual

The participants were also given a user manual to aid in setting up the testbed.

1. *Connect the supplied mobile device to your network and verify that it has an Internet connection.*

2. *Enable a mobile hotspot: Settings -> Network and Internet -> Hotspot and tethering -> Wi-Fi hotspot -> On.*

3. *Open the Cisco AnyConnect application and enable AnyConnect VPN.*

4. *Plug the chargers into power sockets or extension cords. Make sure that they don't obstruct your movements, i.e. you can fully open and close the cupboards, dishwashers, etc.*

5. *Use double-sided stickers to attach Raspberry Pi devices to moving surfaces (e.g. a fridge door, a dishwasher door, or a cupboard door), or floor.*

6. *Attach the contact sensors such that one part is placed on the same moving surface as the Raspberry Pi device and the other part remains on the non-moving surface. The two parts should be close to each other when the doors are closed.*

7. *One leg of the contact sensor should be attached to PIN 20 on the RaspberryPi board, and the other leg to GND.*

8. *Connect RaspberryPi devices to chargers.*

9. *Each device needs approximately **two minutes** to boot. You can verify the progress by checking the LED light on the device (i.e. the flashing green light indicates that the device is still booting while the red light means that it has booted).*

10. *Please fill in the attached Experimental Setup form indicating where the devices are placed.*

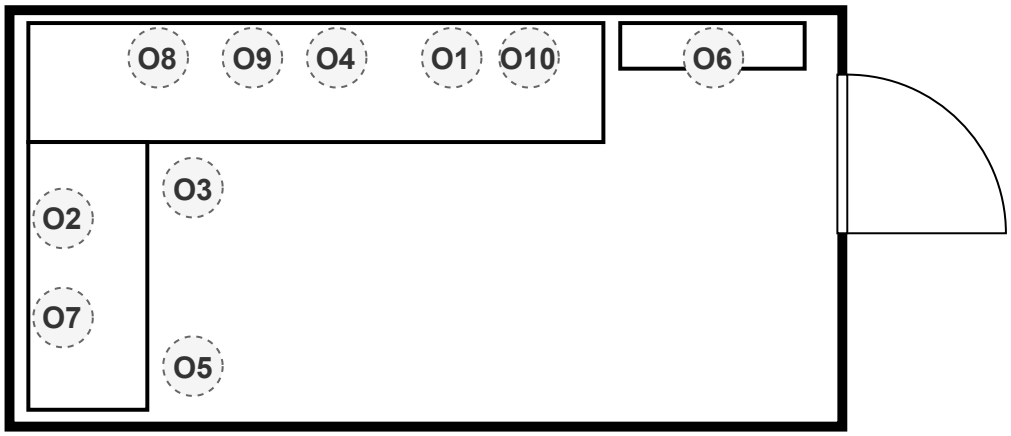
11. *Try opening and closing every door that the sensors are placed on, so you can verify whether the contact sensors are connected properly (i.e. they won't fall off in the middle of your experiment - if they do, please repeat the round).*

12. *Optional*: *Please take pictures of your experimental setup. This will be helpful to understand potential anomalies in the dataset. If you are willing to share the pictures of your experimental setup with us, please send them to the leading researcher. Please indicate in the email if they can be used in the research publications as an example setup.*

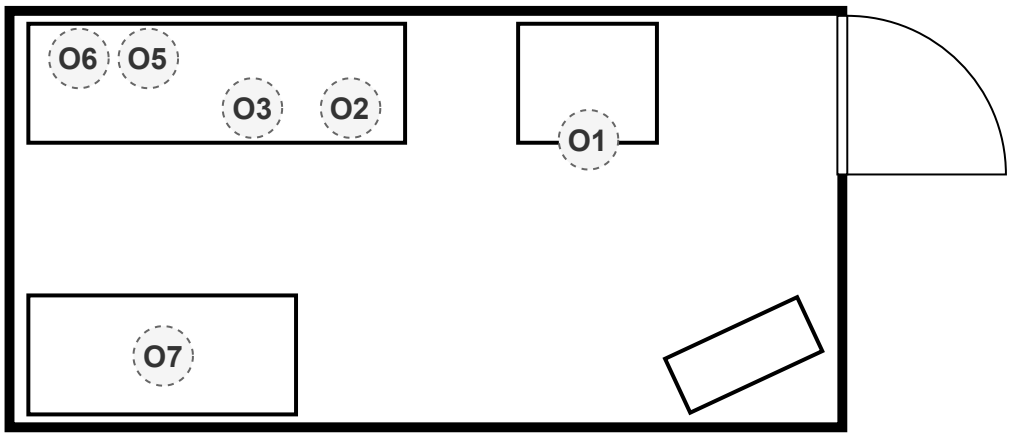
B

House Layouts in User Study

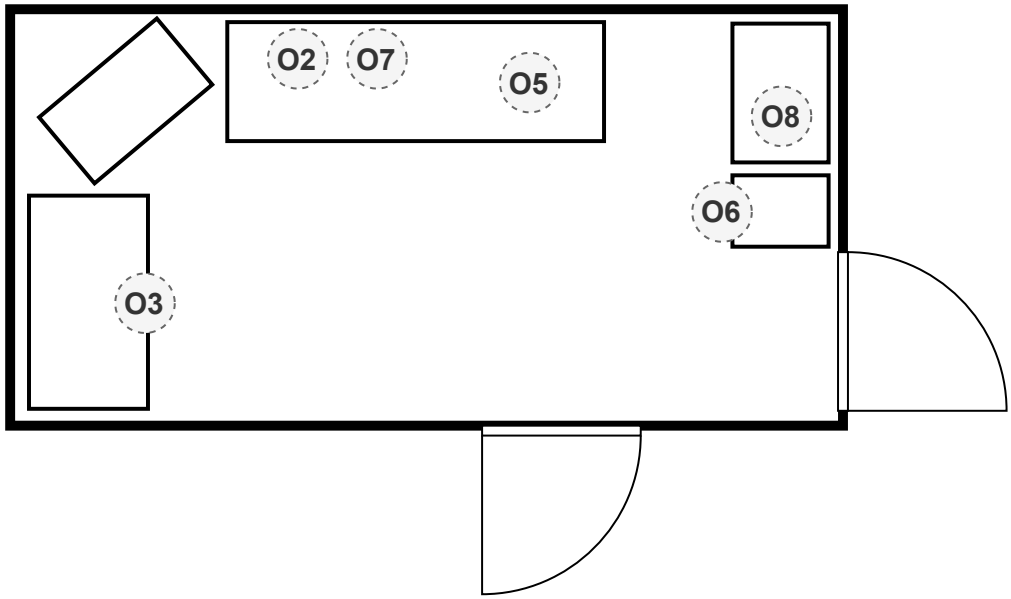
Participants in the first BeeHIVE study in households 1, 4, 5, and 6 shared their house layouts as shown in Fig B.1.



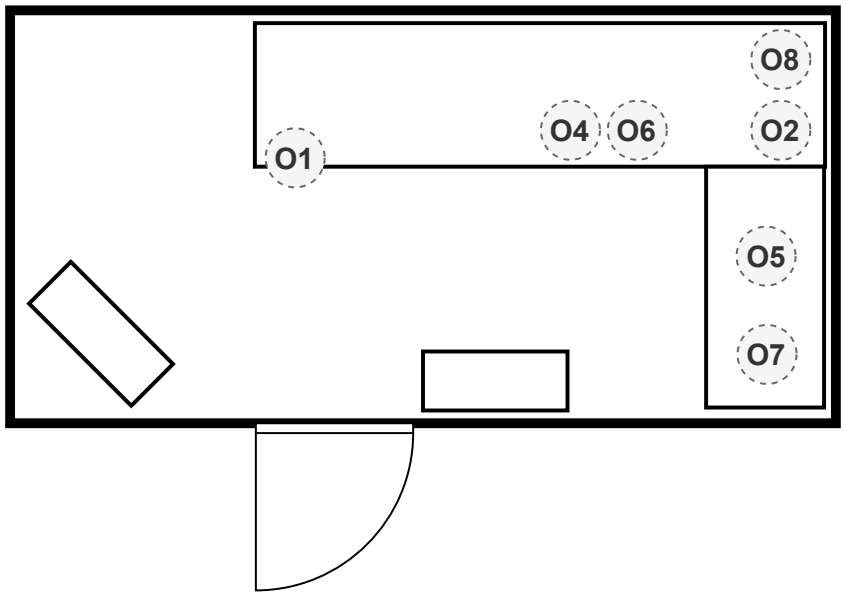
(a) Household 1.



(b) Household 4.



(c) Household 5.



(d) Household 6.

Figure B.1: The household layouts in the user study.

References

[1] Priya Suresh et al. “A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment”. In: *2014 International conference on science engineering and management research (ICSEMR)*. IEEE. 2014, pp. 1–8.

[2] Zhi-Kai Zhang et al. “IoT security: ongoing challenges and research opportunities”. In: *Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on*. IEEE. 2014, pp. 230–234.

[3] Arijit Ukil, Soma Bandyopadhyay, and Arpan Pal. “Privacy for IoT: Involuntary privacy enablement for smart energy systems”. In: *Communications (ICC), 2015 IEEE International Conference on*. IEEE. 2015, pp. 536–541.

[4] Agnieszka Radziwon et al. “The smart factory: exploring adaptive and flexible manufacturing solutions”. In: *Procedia engineering* 69 (2014), pp. 1184–1190.

[5] A Saleema and Sabu M Thampi. “Voice Biometrics: The Promising Future of Authentication in the Internet of Things”. In: *Handbook of Research on Cloud and Fog Computing Infrastructures for Data Science*. IGI Global, 2018, pp. 360–389.

[6] Yan Meng et al. “Securing consumer IoT in the smart home: Architecture, challenges, and countermeasures”. In: *IEEE Wireless Communications* 25.6 (2018), pp. 53–59.

[7] Jagmohan Chauhan et al. “Breathing-based authentication on resource-constrained iot devices using recurrent neural networks”. In: *Computer* 51.5 (2018), pp. 60–67.

[8] Fangmin Sun et al. “Accelerometer-based speed-adaptive gait authentication method for wearable IoT devices”. In: *IEEE Internet of Things Journal* 6.1 (2018), pp. 820–830.

[9] Andrea F Abate et al. “MUBAI: multiagent biometrics for ambient intelligence”. In: *Journal of Ambient Intelligence and Humanized Computing* 2.2 (2011), pp. 81–89.

[10] A. Castiglione et al. “Context Aware Ubiquitous Biometrics in Edge of Military Things”. In: *IEEE Cloud Computing* 4.6 (2017), pp. 16–20.

[11] Dong-Su Kim and Kwang-Seok Hong. “Multimodal biometric authentication using teeth image and voice in mobile environment”. In: *IEEE Transactions on Consumer Electronics* 54.4 (2008), pp. 1790–1797.

[12] Nemanja Maček et al. “Multimodal biometric authentication in IoT: Single camera case study”. In: (2016).

[13] Oscar Olazabal et al. “Multimodal Biometrics for Enhanced IoT Security”. In: *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE. 2019, pp. 0886–0893.

[14] Mikhail Gofman et al. “Multimodal biometrics via discriminant correlation analysis on mobile devices”. In: *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer … 2018, pp. 174–181.

[15] Wenbo Ding and Hongxin Hu. “On the safety of iot device physical interaction control”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 832–846.

[16] Haotian Chi et al. “Cross-app interference threats in smart homes: Categorization, detection and handling”. In: *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE. 2020, pp. 411–423.

[17] Qi Wang et al. “Charting the attack surface of trigger-action IoT platforms”. In: *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 2019, pp. 1439–1453.

[18] Klaudia Krawiecka et al. “Plug-and-Play: Framework for Remote Experimentation in Cyber Security”. In: *European Symposium on Usable Security 2021*. EuroUSEC ’21. Karlsruhe, Germany: Association for Computing Machinery, 2021, pp. 48–58. URL: https://doi.org/10.1145/3481357.3481518.

[19] Klaudia Krawiecka et al. “Biometric Identification System based on Object Interactions in Internet of Things Environments”. In: *2022 IEEE Security and Privacy Workshops (SPW)*. 2022, pp. 215–221.

[20] Arman Bhalla et al. “MoveAR: Continuous Biometric Authentication for Augmented Reality Headsets”. In: *Proceedings of the 7th ACM on Cyber-Physical System Security Workshop*. CPSS ’21. Virtual Event, Hong Kong: Association for Computing Machinery, 2021, pp. 41–52. URL: https://doi.org/10.1145/3457339.3457983.

[21] Fatima Hussain et al. “Machine Learning in IoT Security: Current Solutions and Future Challenges”. In: *CoRR* abs/1904.05735 (2019). arXiv: 1904.05735. URL: http://arxiv.org/abs/1904.05735.

[22] Eric Zeng, Shrirang Mare, and Franziska Roesner. “End user security and privacy concerns with smart homes”. In: *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. 2017, pp. 65–80.

[23] Shrirang Mare et al. “Consumer smart homes: Where we are and where we need to go”. In: *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. 2019, pp. 117–122.

[24] Petar Kocovic et al. “Internet-of-Thing for Consumer Electronics (IoT4CE)”. In: ().

[25] Sunil Manandhar et al. “Towards a natural perspective of smart homes for practical security and safety analyses”. In: *2020 ieee symposium on security and privacy (sp)*. IEEE. 2020, pp. 482–499.

[26] Ejaz Ahmed et al. “Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges”. In: *IEEE Wireless Communications* 23.5 (2016), pp. 10–16.

[27] Jun Han et al. “Do you feel what I hear? Enabling autonomous IoT device pairing using different sensor types”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 836–852.

[28] Richard Harper. *Inside the smart home*. Springer Science & Business Media, 2006.

[29] Scott Davidoff et al. “Principles of smart home control”. In: *International conference on ubiquitous computing*. Springer. 2006, pp. 19–34.

[30] Eric Zeng and Franziska Roesner. “Understanding and Improving Security and Privacy in {Multi-User} Smart Homes: A Design Exploration and {In-Home} User Study”. In: *28th USENIX Security Symposium (USENIX Security 19)*. 2019, pp. 159–176.

[31] Omar Alrawi et al. “Sok: Security evaluation of home-based iot deployments”. In: *2019 IEEE symposium on security and privacy (sp)*. IEEE. 2019, pp. 1362–1380.

[32] Thien Duc Nguyen et al. “DÏoT: A federated self-learning anomaly detection system for IoT”. In: *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*. IEEE. 2019, pp. 756–767.

[33] Artur Marzano et al. “The Evolution of Bashlite and Mirai IoT Botnets”. In: *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2018, pp. 00813–00818.

[34] Constantinos Kolias et al. “DDoS in the IoT: Mirai and other botnets”. In: *Computer* 50.7 (2017), pp. 80–84.

[35] Sam Edwards and Ioannis Profetis. “Hajime: Analysis of a decentralized internet worm for IoT devices”. In: *Rapidity Networks* 16 (2016).

[36] Joseph Bonneau. “The science of guessing: analyzing an anonymized corpus of 70 million passwords”. In: *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE. 2012, pp. 538–552.

[37] Jordi Mongay Batalla, Athanasios Vasilakos, and Mariusz Gajewski. “Secure smart homes: opportunities and challenges”. In: *ACM Computing Surveys (CSUR)* 50.5 (2017), p. 75.

[38] E. Fernandes et al. “Security Implications of Permission Models in Smart-Home Application Frameworks”. In: *IEEE Security Privacy* 15.2 (Mar. 2017), pp. 24–30.

[39] Kristin Williams et al. “The Upcycled Home: Removing barriers to lightweight modification of the home’s everyday objects”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–13.

[40] Shijia Pan et al. “Universense: Iot device pairing through heterogeneous sensing signals”. In: *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*. 2018, pp. 55–60.

[41] Carlos Ruiz et al. “Demo abstract: Secure pairing via video and IMU verification”. In: *2019 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE. 2019, pp. 333–334.

[42] Kashif Habib and Wolfgang Leister. “Context-aware authentication for the internet of things”. In: *Elev. Int. Conf. Auton. Auton. Syst. fined*. 2015, pp. 134–139.

[43] Yosef Ashibani, Dylan Kauling, and Qusay H Mahmoud. “A context-aware authentication framework for smart homes”. In: *Electrical and Computer Engineering (CCECE), 2017 IEEE 30th Canadian Conference on*. IEEE. 2017, pp. 1–5.

[44] Moritz Loske, Lukas Rothe, and Dominik G Gertler. “Context-aware authentication: State-of-the-art evaluation and adaption to the IIoT”. In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE. 2019, pp. 64–69.

[45] Bingxian Lu et al. “LaSa: Location Aware Wireless Security Access Control for IoT Systems”. In: *Mobile Networks and Applications* 24.3 (2019), pp. 748–760.

[46] Amit Kumar Sikder et al. “Aegis: A context-aware security framework for smart home systems”. In: *Proceedings of the 35th Annual Computer Security Applications Conference*. 2019, pp. 28–41.

[47] Aniello Castiglione et al. “Context Aware Ubiquitous Biometrics in Edge of Military Things”. In: *IEEE Cloud Computing* 4.6 (2017), pp. 16–20.

[48] Mauro Conti and Chhagan Lal. “Context-based co-presence detection techniques: A survey”. In: *Computers & Security* 88 (2020), p. 101652.

[49] Simon Birnbach, Simon Eberz, and Ivan Martinovic. “Peeves: Physical Event Verification in Smart Homes”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 1455–1467. URL: https://doi.org/10.1145/3319535.3354254.

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

[51] Earlence Fernandes et al. “Decentralized action integrity for trigger-action IoT platforms”. In: *Proceedings 2018 Network and Distributed System Security Symposium*. 2018.

[52] Hang Hu et al. “Security vetting process of smart-home assistant applications: A first look and case studies”. In: *arXiv preprint arXiv:2001.04520* (2020).

[53] Pengfei Peng and An Wang. “SmartMon: Misbehavior Detection via Monitoring Smart Home Automations”. In: *2020 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE. 2020, pp. 327–333.

[54] Dang Tu Nguyen et al. “IotSan: Fortifying the safety of IoT systems”. In: *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*. 2018, pp. 191–203.

[55] Rahmadi Trimananda et al. “Understanding and automatically detecting conflicting interactions between smart home IoT applications”. In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020, pp. 1215–1227.

[56] Qi Wang et al. “Fear and logging in the internet of things”. In: *Network and Distributed Systems Symposium*. 2018.

[57] Jingwen Fan et al. “Ruledger: Ensuring execution integrity in trigger-action IoT platforms”. In: *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE. 2021, pp. 1–10.

[58] Roman V Yampolskiy. “Mimicry attack on strategy-based behavioral biometric”. In: *Fifth International Conference on Information Technology: New Generations (itng 2008)*. IEEE. 2008, pp. 916–921.

[59] Hugo B Barra. *Voice authentication and command*. US Patent 8,543,834. Sept. 2013.

[60] Logan Blue et al. “2ma: Verifying voice commands via two microphone authentication”. In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 2018, pp. 89–100.

[61] Nicholas Carlini et al. “Hidden voice commands”. In: *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 2016, pp. 513–530.

[62] Guoming Zhang et al. “Dolphinattack: Inaudible voice commands”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 103–117.

[63] Wenrui Diao et al. “Your voice assistant is mine: How to abuse speakers to steal information and control your phone”. In: *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*. 2014, pp. 63–74.

[64] Nan Zhang et al. “Understanding and mitigating the security risks of voice-controlled third-party skills on amazon alexa and google home”. In: *arXiv preprint arXiv:1805.01525* (2018).

[65] Pratik Musale, Duin Baek, and Bong Jun Choi. “Lightweight gait based authentication technique for IoT using subconscious level activities”. In: *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*. IEEE. 2018, pp. 564–567.

[66] Pratik Musale et al. “You walk, we authenticate: Lightweight seamless authentication based on gait in wearable IoT systems”. In: *IEEE Access* 7 (2019), pp. 37883–37895.

[67] Samera Batool, Nazar A Saqib, and Muazzam A Khan. “Internet of Things data analytics for user authentication and activity recognition”. In: *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE. 2017, pp. 183–187.

[68] Antonio A Aguileta et al. “Multi-Sensor Fusion for Activity Recognition—A Survey”. In: *Sensors* 19.17 (2019), p. 3808.

[69] Piyush K Sharma, Mark Dennison, and Adrienne Raglin. “Iot solutions with multi-sensor fusion and signal-image encoding for secure data transfer and decision making”. In: *arXiv preprint arXiv:2106.01497* (2021).

[70] Ashraful Alam et al. “Leveraging Sensor Fusion and Sensor-Body Position for Activity Recognition for Wearable Mobile Technologies.” In: *International Journal of Interactive Mobile Technologies* 15.17 (2021).

[71] Md Shahriar Tasjid and Ahmed Al Marouf. “Leveraging Smartphone Sensors for Detecting Abnormal Gait for Smart Wearable Mobile Technologies”. In: *iJIM* 15.24 (2021), p. 167.

[72] F. Montalto et al. “MuSA: Wearable Multi Sensor Assistant for Human Activity Recognition and Indoor Localization”. In: vol. 11. July 2015, pp. 81–92.

[73] Pranjal Kumar and Siddhartha Chauhan. “Human activity recognition with deep learning: overview, challenges and possibilities”. In: *CCF Transactions on Pervasive Computing and Interaction* (2021), pp. 1–29.

[74] Raffaele Gravina and Qimeng Li. “Emotion-relevant activity recognition based on smart cushion using multi-sensor fusion”. In: *Information Fusion* 48 (2019), pp. 1–10.

[75] Elakkiya Ellavarason et al. “Touch-dynamics based Behavioural Biometrics on Mobile Devices–A Review from a Usability and Performance Perspective”. In: *ACM Computing Surveys (CSUR)* 53.6 (2020), pp. 1–36.

[76] Shurook S Almohamade, John A Clark, and James Law. “Behaviour-Based Biometrics for Continuous User Authentication to Industrial Collaborative Robots”. In: *International Conference on Information Technology and Communications Security*. Springer. 2020, pp. 185–197.

[77] Akriti Verma, Valeh Moghaddam, and Adnan Anwar. “Data-driven behavioural biometrics for continuous and adaptive user verification using Smartphone and Smartwatch”. In: *arXiv preprint arXiv:2110.03149* (2021).

[78] Pin Shen Teh et al. “A survey on touch dynamics authentication in mobile devices”. In: *Computers & Security* 59 (2016), pp. 210–235.

[79] Amir E Sarabadani Tafreshi, Sara C Sarabadani Tafreshi, and Amirehsan Sarabadani Tafreshi. “TiltPass: using device tilts as an authentication method”. In: *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. 2017, pp. 378–383.

[80] Wei-Han Lee et al. “Secure pick up: Implicit authentication when you start using the smartphone”. In: *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies*. 2017, pp. 67–78.

[81] Attaullah Buriro et al. “Hold and Sign: A Novel Behavioral Biometrics for Smartphone User Authentication”. In: *2016 IEEE Security and Privacy Workshops (SPW)*. 2016, pp. 276–285.

[82] Jun Han et al. “Smart home occupant identification via sensor fusion across on-object devices”. In: *ACM Transactions on Sensor Networks (TOSN)* 14.3-4 (2018), pp. 1–22.

[83] Roman V. Yampolskiy. “Mimicry Attack on Strategy-Based Behavioral Biometric”. In: *Fifth International Conference on Information Technology: New Generations (itng 2008)*. 2008, pp. 916–921.

[84] Akshay Agarwal et al. “Face presentation attack with latex masks in multispectral videos”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 81–89.

[85] Christian Rathgeb, Pawel Drozdowski, and Christoph Busch. “Makeup presentation attacks: Review and detection performance benchmark”. In: *IEEE Access* 8 (2020), pp. 224958–224973.

[86] Esther Shein. “Using makeup to block surveillance”. In: *Communications of the ACM* 65.7 (2022), pp. 21–23.

[87] Raghavendra Ramachandra and Christoph Busch. “Presentation attack detection methods for face recognition systems: A comprehensive survey”. In: *ACM Computing Surveys (CSUR)* 50.1 (2017), pp. 1–37.

[88] Morgan Frearson and Kien Nguyen. “Adversarial attack on facial recognition using visible light”. In: *arXiv preprint arXiv:2011.12680* (2020).

[89] Zhe Zhou et al. “Invisible mask: Practical attacks on face recognition with infrared”. In: *arXiv preprint arXiv:1803.04683* (2018).

[90] Xiaopeng Li et al. “Touch well before use: Intuitive and secure authentication for iot devices”. In: *The 25th annual international conference on mobile computing and networking*. 2019, pp. 1–17.

[91] Ajita Rattani and Norman Poh. “Biometric system design under zero and non-zero effort attacks”. In: *2013 International Conference on Biometrics (ICB)*. IEEE. 2013, pp. 1–8.

[92] Emanuela Marasco, Mohamed Shehab, and Bojan Cukic. “A Methodology for Prevention of Biometric Presentation Attacks”. In: *2016 Seventh Latin-American Symposium on Dependable Computing (LADC)*. 2016, pp. 9–14.

[93] Aditya Sundararajan, Arif I Sarwat, and Alexander Pons. "A survey on modality characteristics, performance evaluation metrics, and security for traditional and wearable biometric systems". In: *ACM Computing Surveys (CSUR)* 52.2 (2019), pp. 1–36.

[94] Precise Biometrics. "Understanding biometric performance evaluation". In: *URL: https://precisebiometrics.com/wp-content/uploads/2014/11/White-Paper-Understanding-Biometric-Performance-Evaluation.pdf (pristupljeno: srpanj 2018.)[10]* (2014).

[95] C.M. Dayton, ERIC Clearinghouse on Assessment, and Evaluation. *Some Key Concepts for the Design and Review of Empirical Research*. ERIC digest. ERIC Clearinghouse on Assessment and Evaluation, 2002. URL: https://books.google.co.uk/books?id=A9IeuwEACAAJ.

[96] Donald T. Campbell and Julian C. Stanley. *Experimental and quasi-experimental designs for research*. Ravenio Books, 2015.

[97] Internet School Experimental System (iSES). *Collection of examples for remote laboratories*. 2018. URL: https://www.ises.info/index.php/en/laboratory (visited on 05/28/2021).

[98] Dartmouth College. *Teach Remotely: How to Teach From Anywhere*. 2020. URL: https://sites.dartmouth.edu/teachremote/remote-lab-activities-and-experiences (visited on 05/28/2021).

[99] Milica B Naumovic and Dragan Zivanovic. "Remote Experiments in Control Engineering Education Laboratory". In: *Int. J. Online Eng.* 4.2 (2008), pp. 48–53.

[100] Christian Thomsen, Harald Scheel, and Sabine Morgner. "Remote experiments in experimental physics". In: *Proceedings of the ISPRS E-Learning*. Vol. 2005. Citeseer. 2005.

[101] Tomáš Kozıék and Marek Šimon. "Preparing and managing the remote experiment in education". In: *2012 15th International Conference on Interactive Collaborative Learning (ICL)*. IEEE. 2012, pp. 1–4.

[102] Kendall Powell. "Science-ing from home". In: *Nature* 580.7802 (2020), pp. 419–422.

[103] R. Ahuja et al. "Protecting against Inference Attacks on Co-Location Data". In: *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. July 2019, pp. 1–11.

[104] K. Lin et al. "Enhanced Fingerprinting and Trajectory Prediction for IoT Localization in Smart Buildings". In: *IEEE Transactions on Automation Science and Engineering* 13.3 (July 2016), pp. 1294–1307.

[105] Eric Zeng and Franziska Roesner. "Understanding and Improving Security and Privacy in Multi-User Smart Homes: A Design Exploration and In-Home User Study". In: *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 159–176. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/zeng.

[106] Ashraf Alkhresheh et al. "Dynamic Access Control Framework for Internet of Things". PhD thesis. 2019.

[107] Jun Han et al. “Do you feel what I hear? Enabling autonomous IoT device pairing using different sensor types”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 836–852.

[108] N. Saparkhojayev et al. “NFC-enabled access control and management system”. In: *2014 International Conference on Web and Open Access to Learning (ICWOAL)*. Nov. 2014, pp. 1–4.

[109] Y. Chang et al. “Toward a NFC Phone-Driven Context Awareness Smart Environment”. In: *2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*. July 2009, pp. 298–303.

[110] Avgoustinos Filippoupolitis et al. “Location-enhanced activity recognition in indoor environments using off the shelf smart watch technology and BLE beacons”. In: *Sensors* 17.6 (2017), p. 1230.

[111] Bryan Parno, Adrian Perrig, and Virgil D Gligor. “Pico: no more passwords!” In: *Proceedings of the 2011 IEEE Symposium on Security and Privacy*. IEEE. 2011.

[112] Jonathan M McCune, Adrian Perrig, and Michael K Reiter. “Seeing-is-believing: Using camera phones for human-verifiable authentication”. In: *Security and privacy, 2005 IEEE symposium on*. IEEE. 2005, pp. 110–124.

[113] Arun Kumar et al. “A comparative study of secure device pairing methods”. In: *Pervasive and Mobile Computing* 5.6 (2009), pp. 734–749.

[114] Nitesh Saxena et al. “Secure device pairing based on a visual channel”. In: *Security and Privacy, 2006 IEEE Symposium on*. IEEE. 2006, 6–pp.

[115] Dongsik Jo and Gerard Jounghyun Kim. “ARIoT: scalable augmented reality framework for interacting with Internet of Things appliances everywhere”. In: *IEEE Transactions on Consumer Electronics* 62.3 (2016), pp. 334–340.

[116] Esha Nerurkar, Simon Lynen, and Sheng Zhao. *System and method for concurrent odometry and mapping*. US Patent App. 15/595,617. Nov. 2017.

[117] Serge Vaudenay. “Secure Communications over Insecure Channels Based on Short Authenticated Strings”. In: *Advances in Cryptology – CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005. Proceedings*. Ed. by Victor Shoup. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 309–326.

[118] *Device Pairing Using Short Authentication Strings*. `https://tools.ietf.org/id/draft-ietf-dnssd-pairing-02.html`. 2017.

[119] Danny Dolev and Andrew Yao. “On the security of public key protocols”. In: *IEEE Transactions on information theory* 29.2 (1983), pp. 198–208.

[120] Nitesh Saxena and Md Borhan Uddin. “Secure pairing of “Interface-constrained” devices resistant against rushing user behavior”. In: *International Conference on Applied Cryptography and Network Security*. Springer. 2009, pp. 34–52.

[121] *Google ARCore*. `https://developers.google.com/ar/`. 2017.

[122] Inc. Initial State Technologies. *Your Very Own Raspberry Pi Smart Home Hub*. 2016. URL: `https://github.com/initialstate/node-hub/wiki`.

[123] Michael O Jewell, Enrico Costanza, and Jacob Kittley-Davies. "Connecting the things to the internet: an evaluation of four configuration strategies for wi-fi devices with minimal user interfaces". In: *Proceedings of the 2015 ACM International Joint Conference on pervasive and ubiquitous computing*. ACM. 2015, pp. 767–778.

[124] The Royal Society. *Science in Lockdown: the effects of COVID-19 on research and researchers*. 2020. URL: https://royalsociety.org/blog/2020/05/science-in-lockdown-part-one (visited on 05/27/2021).

[125] Edgar Padilla, Jaime C. Acosta, and Christopher D. Kiekintveld. "Cybersecurity Methodology for Specialized Behavior Analysis". In: *Digital Forensics and Cyber Crime*. Ed. by Sanjay Goel et al. Cham: Springer International Publishing, 2021, pp. 237–243.

[126] Carrie Gardner et al. *Using Camouflaged Cyber Simulations as a Model to Ensure Validity in Cybersecurity Experimentation*. 2019. arXiv: 1905.07059 [cs.CR].

[127] Aditya Ashok et al. "A Multi-level Fidelity Microgrid Testbed Model for Cybersecurity Experimentation". In: *12th USENIX Workshop on Cyber Security Experimentation and Test (CSET 19)*. 2019.

[128] David Balenson, Laura Tinnel, and T Benzel. "Cybersecurity experimentation of the future (CEF): catalyzing a new generation of experimental cybersecurity research". In: *SRI International, Tech. Rep.* (2015).

[129] Xu Sun and Andrew May. "A comparison of field-based and lab-based experiments to evaluate user experience of personalised mobile devices". In: *Advances in Human-Computer Interaction* 2013 (2013).

[130] Jesper Kjeldskov and Mikael B Skov. "Was it worth the hassle? Ten years of mobile HCI research discussions on lab and field evaluations". In: *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*. 2014, pp. 43–52.

[131] Hassan A Aziz. "Comparison between field research and controlled laboratory research". In: *Archives of Clinical and Biomedical Research* 1.2 (2017), pp. 101–104.

[132] Katta Spiel et al. "Nothing About Us Without Us: Investigating the Role of Critical Disability Studies in HCI". In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI EA '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–8. URL: https://doi.org/10.1145/3334480.3375150.

[133] Heather Becker et al. "Recruiting People With Disabilities as Research Participants: Challenges and Strategies to Address Them". In: *Mental retardation* 42 (Jan. 2005), pp. 471–5.

[134] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. "MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks". In: *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*. IEEE. 2008, pp. 791–798.

[135] Roger A Light. "Mosquitto: server and client implementation of the MQTT protocol". In: *Journal of Open Source Software* 2.13 (2017), p. 265.

[136] Benjamin K. Sovacool and Dylan D. Furszyfer Del Rio. "Smart home technologies in Europe: A critical review of concepts, benefits, risks and policies". In: *Renewable and Sustainable Energy Reviews* 120 (2020), p. 109663. URL: http://www.sciencedirect.com/science/article/pii/S1364032119308688.

[137] Alex Barros et al. "Heart of IoT: ECG as biometric sign for authentication and identification". In: *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE. 2019, pp. 307–312.

[138] Wencheng Yang et al. "Biometrics for Internet-of-Things Security: A Review". In: *Sensors* 21.18 (2021), p. 6163.

[139] Naomi Irvine et al. "Neural Network Ensembles for Sensor-Based Human Activity Recognition Within Smart Environments". In: *Sensors* 20.1 (2020). URL: https://www.mdpi.com/1424-8220/20/1/216.

[140] Hassan Khan, Urs Hengartner, and Daniel Vogel. "Augmented reality-based mimicry attacks on behaviour-based smartphone authentication". In: *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 2018, pp. 41–53.

[141] Samanta Rosati, Gabriella Balestra, and Marco Knaflitz. "Comparison of different sets of features for human activity recognition by wearable sensors". In: *Sensors* 18.12 (2018), p. 4189.

[142] Mario Beraha et al. *Feature Selection via Mutual Information: New Theoretical Insights*. 2019. arXiv: 1907.07384 [cs.LG].

[143] David H Wolpert. "Stacked generalization". In: *Neural networks* 5.2 (1992), pp. 241–259.

[144] Enrique Garcia-Ceja, Carlos E. Galván-Tejada, and Ramon Brena. "Multi-view stacking for activity recognition with sound and accelerometer data". In: *Information Fusion* 40 (2018), pp. 45–56. URL: http://www.sciencedirect.com/science/article/pii/S1566253516301932.

[145] Wouter van Loon et al. "Stacked penalized logistic regression for selecting views in multi-view learning". In: *Information Fusion* 61 (Sept. 2020), pp. 113–123. URL: http://dx.doi.org/10.1016/j.inffus.2020.03.007.

[146] Z Berkay Celik, Gang Tan, and Patrick D McDaniel. "IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT." In: *NDSS*. 2019.

[147] Wenbo Ding, Hongxin Hu, and Long Cheng. "IOTSAFE: Enforcing safety and security policy with real IoT physical interaction discovery". In: *the 28th Network and Distributed System Security Symposium (NDSS 2021)*. 2021.

[148] Anil Seth. "Granger causality". In: *Scholarpedia* 2.7 (2007), p. 1667.

[149] Xuegeng Mao and Pengjian Shang. "Transfer entropy between multivariate time series". In: *Communications in Nonlinear Science and Numerical Simulation* 47 (2017), pp. 338–347.

[150] Patricia Wollstadt et al. "IDTxl: The Information Dynamics Toolkit xl: a Python package for the efficient analysis of multivariate information dynamics in networks". In: *arXiv preprint arXiv:1807.10459* (2018).

[151] Aidan Hogan et al. "Knowledge graphs". In: *ACM Computing Surveys (CSUR)* 54.4 (2021), pp. 1–37.

[152] Aric Hagberg, Pieter Swart, and Daniel S Chult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[153] Furkan Goksel et al. "On the safety implications of misordered events and commands in IoT systems". In: *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE. 2021, pp. 235–241.