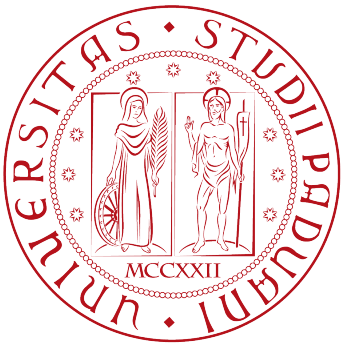


University of Padua
Human Inspired Technology Research Centre
Curriculum: Computer Science and Innovation for Societal Challenges

---

Human-Computer Interaction: Security Aspects

Candidate
QianQian Li

Supervisor
Prof. Mauro Conti

Co-Supervisor
Prof. Luciano Gamberini

University of Padova, Italy

October 31st, 2017

Acknowledgments

In order to challenge my interest in the research and know more deeply about myself, I applied the Ph.D. at the University of Padova and thanks the CARIPARIO scholarship for giving me the chance to chanllenge myself. After this three years, I want to express my thankfulness as follows:

First of all, I would like to thank my advisor Prof. Mauro Conti for his guidance. My thankfulness to him is expressed in three aspects.

- Under the condition of unfamiliar environment, unfamiliar English, it was hard to start the first year of my Ph.D. However, I was greatly encouraged by my advisor's patient encouragement and guidance. He tried to use all of his patience to tell me how to be a professional researcher from the aspects of attending courses and seminars, carrying on research activities, and writing scientific papers. More than that, he even told me which should take and report for a meeting. He told me how to update my research plan, report, weekly update, even a punctuation.

- In order to carry on the interdisciplinary topic, he provided me the chance to collaborate with Riccardo and Merylin. Moreover, he encouraged me to go University of Florida and ShanDong Normal University for a visiting and established a collaboration. During the collaboration, the collaborators gave me more motivation and I really learned a lot from other researchers that collaborated and visited our research group (particularly with Reza Mohammadi). I started to enjoy the challenge in the research and improved a lot of my work on Software Defined Networking and Wireless Sensor Network.

- When I reported my work, my supervisor always encourages me and supports me. His meticulous and rigorous attitude, enthusiasm, work efficiency really inspired me. It is really my honor to know him and he will be always my example in my life.

Second, I want to sincerely express my thankfulness to Prof. Andy Li, Prof. Chuanhuang Li and Prof. FangAi Liu of giving me the inspiration for my research. They gave me the insightful and valuable comments and suggestions for my paper, and how to be a professional researcher.

Third, I would like to express my great thankfulness to my colleagues (Reza Khosh Kangini, Riccardo Spolaor, Tran van Dinh, Merylin Monaro) who attended courses together in Brain, Mind and Computer Science. All of you are so nice and kind to give me the courage to face something I am not dare to. Not only they encouraged to speak English and speak boldly my opinions, but also they helped me improve my annual presentation by telling me their own experience which is really useful for my following presentation on ARES'17 and ICCP'17 conferences. Thanks very much for their advice also for my topic.

Fourth, I really would like to thanks my officemates and SPRITZ colleagues (Tran van Dinh, Muhammad Hassan Raza Khan, Rabbani MD Masoom, Giuseppe Cascavilla, Yan Hu, Ankit Gangwal, Zahra Pooranian, Chhagan Lal, Sarada Prasad Gochhayat, Moreno Ambrosin). I really enjoyed the time in office Room 732 of Torre Archimede. They are amazing friends to talk, to release my burden, enjoy my happiness and inspire my idea of research. I really learn from everyone of them. I am really happy that all of you accompanied me during my precious Ph.D. time and I will never forget the happy and sad time with you.

Fifth, I would like to thanks the Chinese friends here in Padova, especially my roommate (Meng Zheng). They made my life abroad more interesting, happy and beautiful. Their help for me is something I will never forget and I am appreciated that for the rest of my life.

Sixth, I am grateful to my colleague and boyfriend Riccardo Spolaor. The encouragement, confidence, tenderness, and instruction he gave to me helped me overcome the most difficult time during my Ph.D. His patience and kindness not only helped my research, but also gave me more happiness in my life. I started to be more optimistic with my research. I will become a better myself to deserve your help.

Seventh, a huge thank goes to my family, that always supported me in their life, especially these three years. Words cannot express how grateful I am to my family They were so worried about my life and study in Italy, especially my father. They are too traditional to express their love and worry for you. It is my first time to experience an older traditional farmer father to express his love by word. Moreover, it is their love and support to help me and accompany my Ph.D. Thanks for their support, understanding, and accompanying.

Last but not the least, I would like to be pride to thank myself. For these three years, I had experienced what I had never thought to go through. At first, I refused to accept the change, and now I am happy to accept the challenge. I really appreciate all the experience which helps me to improve

myself and know more clearly about myself. These three years make a new and good myself. Please allow me to be proud of myself and say "thanks for your effort and perseverance".

QianQian Li
Padova, Oct 31st, 2017

Abstract

Along with the rapid development of intelligent information age, users are having a growing interaction with smart devices. Such smart devices are interconnected together in the Internet of Things (IoT). The sensors of IoT devices collect information about users' behaviors from the interaction between users and devices. Since users interact with IoT smart devices for the daily communication and social network activities, such interaction generates a huge amount of network traffic. Hence, users' behaviors are playing an important role in the security of IoT smart devices, and the security aspects of Human-Computer Interaction are becoming significant.

In this dissertation, we provide a threefold contribution: (1) we review security challenges of HCI-based authentication, and design a tool to detect deceitful users via keystroke dynamics; (2) we present the impact of users' behaviors on network traffic, and propose a framework to manage such network traffic; (3) we illustrate a proposal for energy-constrained IoT smart devices to be resilient against energy attack and efficient in network communication.

More in detail, in the first part of this thesis, we investigate how users' behaviors impact on the way they interact with a device. Then we review the work related to security challenges of HCI-based authentication on smartphones, and Brain-Computer Interfaces (BCI). Moreover, we design a tool to assess the truthfulness of the information that users input using a computer keyboard. This tool is based on keystroke dynamics and it relies on machine learning technique to achieve this goal. To the best of our knowledge, this is the first work that associates the typing users' behaviors with the production of deceptive personal information. We reached an overall accuracy of 76% in the classification of a single answer as truthful or deceptive.

In the second part of this thesis, we review the analysis of network traffic, especially related to the interaction between mobile devices and users. Since the interaction generates a huge amount of network traffic, we propose an innovative framework, GolfEngine, to manage and control the impact

of users behavior on the network relying on Software Defined Networking (SDN) techniques. GolfEngine provides users a tool to build their security applications and offers Graphical User Interface (GUI) for managing and monitoring the network. In particular, GolfEngine provides the function of checking policy conflicts when users design security applications and the mechanism to check data storage redundancy. GolfEngine not only prevents the malicious inputting policies but also it enforces the security about network management of network traffic. The results of our simulation underline that GolfEngine provides an efficient, secure, and robust performance for managing network traffic via SDN.

In the third and last part of this dissertation, we analyze the security aspects of battery-equipped IoT devices from the energy consumption perspective. Although most of the energy consumption of IoT devices is due to user interaction, there is still a significant amount of energy consumed by point-to-point communication and IoT network management. In this scenario, an adversary may hijack an IoT device and conduct a Denial of Service attack (DoS) that aims to run out batteries of other devices. Therefore, we propose EnergIoT, a novel method based on energetic policies that prevent such attacks and, at the same time, optimizes the communication between users and IoT devices, and extends the lifetime of the network. EnergIoT relies on a hierarchical clustering approach, based on different duty cycle ratios, to maximize network lifetime of energy-constrained smart devices. The results show that EnergIoT enhances the security and improves the network lifetime by 32%, compared to the earlier used approach, without sacrificing the network performance (i.e., end-to-end delay).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human-Computer Interaction (HCI) studies the interaction between human and computer [1]. Research in the area of HCI is fascinating and complex. It is fascinating because there are so many interesting questions and so many challenges. It is complex because it involves human beings, whose behavior is difficult to observe. In the research work presented in this thesis, we focus on an interdisciplinary topic in HCI, called computer science-based HCI [2], especially the security aspect of computer science-based HCI. We deal with user authentication on smart devices relying on the network traffic which is generated during the interaction between users and devices. Before diving into the content of the thesis and its contributions, in this chapter, we first introduce the research motivation and contribution, then we list the publication during my Ph.D.

1.1 Research Motivation and Contribution

The topic of HCI becomes quite hot and interesting for researchers since it involves new technologies. In the ubiquitous computing and smart technology age, smart devices are developing extremely fast and spreading all around of our lives. Consequently, the interaction between users and smart devices becomes more and more popular. Such interaction generates lots of network traffic which can be used for analyzing users' behavior and authenticating users. Moreover, with the growing number of mobile devices and IoT devices, a huge amount of network traffic is transmitted on the network. We try to manage network traffic generated by HCI efficiently and securely through a new paradigm Software Defined Networking (SDN). Finally, given the scenario where an attacker conducts a DoS attack on energy-constrained devices and run out of the battery, it is necessary to manage and extend

the lifetime of communication among HCI related devices efficiently and securely.

The research work presented in this dissertation includes several contributions to the state-of-the-art of HCI security: (i) a tool for detecting deceptive users via keystroke dynamics when users inputting through computer keyboards; (ii) a framework to manage the HCI-related network traffic more securely and efficiently via Software Defined Networking (SDN); and (iii) a method for managing and optimize the energy consumption of a network of battery-powered HCI devices to avoid energy attacks. We describe our work in the following three main logical parts.

- *Security Issues in User Interaction with Devices*, studying the authentication methods for smartphones and Brain-Computer Interfaces, and especially designing a tool based on keystroke dynamics to infer deceptive information.
- *HCI impact on network traffic*, focusing on a survey of HCI-related network traffic analysis on mobile devices, and designing a secure and efficient architecture for managing such network traffic.
- *Energy efficient communication for IoT devices*, proposing a method for avoiding energy to be maliciously consumed and efficiently optimizing the energy consumption of point to point communication.

In what follows, we briefly introduce each of the above parts, and summarize our contribution. Besides, in this dissertation, some figures are re-used and some essays are quoted from the work [3, 4, 5, 6, 7, 8], for which the author of this thesis is co-author.

1.1.1 Security Issues in User Interaction with Devices

Recently, users continuously carry smart devices and use them for daily communication activities and social network interactions. Moreover, users also manage personal data and handle private communications on smart devices. Hence, it is necessary to use authentication methods to prevent from the access of unauthorized users. The most used authentication methods on mobile devices are based on the main secret (e.g., PIN, Graphical pattern, Password, Fingerprint). However, they are not robust against some threats. For example, from shoulder surfing [9] and smudge attacks [10], we can easily recover the PIN and Graphical pattern. Even the knowledge of the secret does not mean that you are legit users. Moreover, the fingerprint can be easily faked. In order to cope with the shortcomings of authentication for mobile devices, researchers proposed a new method called user behavioral biometric authentication [11] that is more robust against the threats.

In the following, we first review HCI-based authentication for smartphones and the categorization of user behavioral biometric authentication.

Then, we review the future of HCI (e.g., Brain-Computer Interfaces) which provides the communication and activities between the devices and users' brain. We study the security challenges in Brain-Computer Interface, in particular, the aspect related to user authentication. Finally, we design a specific tool based on dynamic keystrokes to detect the input of deceptive information.

HCI-based Authentication for Smartphones

Nowadays, users start to develop a symbiotic bound with their smartphones. On the one hand, users continuously carry smartphones with them and use them for daily communication activities and social network interactions. On the other hand, smartphones are enabled with sensors that are able to infer not only information about the context (e.g., location) but also about its owner. Since smartphones handle a huge amount of private information, recent techniques rely on those sensing capabilities to authenticate the user, measuring her biometric features. Instead of relying on a secret, biometric authentication methods rely on physiological and behavioral characteristics of the user.

Contribution: We review the most relevant biometric authentication methods on smartphones proposed in the literature. We categorize the HCI-based authentication methods by the nature of biometrics used, by their temporal features and by the sensing capabilities they rely on. Moreover, we divide the behavioral authentication methods into two categories according to the number of times required to collect data from users and authenticate them. Finally, we draw some future directions in this promising research topic. Our review gives a very clear map for new researchers who want to step into this field and do further study of user behavior authentication on smartphones. This review is illustrated in Chapter 2.

HCI-based Security Challenges in Brain-Computer Interfaces

In order to help disabled people to improve their daily life and communication with other people, it is necessary to build a direct communication between human brain and computer. This is known as Brain-Computer Interfaces (BCI) [12]. BCI systems are deployed to manipulate the brain activity to produce signals that can be used to control computers or communication devices. BCI is becoming popular in medical and non-medical areas. Unfortunately, manufacturers of BCI devices focus on application development, without paying much attention to security and privacy issues. Indeed, an increasing number of attacks on BCI applications underline the existence of such issues. For example, malicious developers of third-party

applications could extract private information of users. Therefore, the robust authentication on BCI is foremost important.

Contribution: We are the first to review security and privacy challenges for BCI applications. In particular, we classify BCI applications into four usage scenarios: 1) neuromedical applications, 2) user authentication, 3) gaming and entertainment, and 4) smartphone-based applications. For each usage scenario, we discuss security and privacy issues and possible countermeasures. Especially, in the user authentication scenario, we survey the work that leverages the brain activities as a special HCI-based authentication method. We believe that our classification will help new researchers in this field to foster future research trends about user authentication for BCI. This review is illustrated in Chapter 3.

Detecting Deceitful Users via Keystroke Dynamics

In the last ten years, keystroke dynamics has been widely used as a biometric measure for user authentication and user identification [13]. However, in order to be able to recognize her, it requires a certain level of knowledge about the alleged user and a specific training. For this reason, we focus on a method that exploits the complexity of the cognitive mechanisms associated with the deceptive production to analyze the user responses. Since this method is based on the general functioning of the human cognitive system, it is possible to distinguish between fake and truthful personal information provided by a user without any prior knowledge about the user who is declaring data. The application of keystroke dynamics to differentiate truthful and deceptive statements is a new field of study. So far, a very few number of works in literature has analyzed the keystroke dynamics pattern associated with the deception production [14, 15]. As far as we know, none has investigated the production of lies regarding of personal information. We believe that a keystroke based deception detection tool could have great potential for the application in online contexts, with a large number of purpose as online baking security, prevention of FPA and so on. Moreover, it may be administered quickly and remotely. The lie detection purpose is implicit, so the typing rhythm can be used as a covert index to revealing deceptive behaviors [16]. Finally, while in other traditional tools for lie detection based on human cognitive functioning the object of the falsehood is predetermined, keystroke analysis offers the possibility to investigate free text contents.

Contribution: We propose a novel method, based on keystroke dynamics, to distinguish between deceptive and truthful personal information when users interact with a computer keyboard. Our method does not need any prior knowledge about the user who is providing data. To our knowledge, this is the first work that associates the typing human behavior with the

production of lies regarding personal information. Via experimental analysis, we assess that this method is able to distinguish between truth and lies on specific types of autobiographical information, with an accuracy higher than 75%. Specifically, for information usually required in online registration forms (e.g., name, surname and email), the typing behavior diverged significantly between truthful or deceptive answers. According to our results, keystroke analysis could have a great potential in detecting the veracity of self-declared information, and it could be applied to a large number of practical scenarios requiring users to input personal data remotely via keyboards. We present this tool in Chapter 4.

1.1.2 HCI Impact on Network Traffic

Since users interact with smart devices for the daily communication and social network activities, human interaction generates a huge amount of network traffic. Moreover, through analyzing the network traffic by machine learning, it can be widely used for authenticating users on mobile devices. Hence, in this part, we talk about the HCI impact on network traffic. Especially, first, we survey all the literature related to network traffic analysis on mobile devices. Then we design an innovative framework to manage such huge amount traffic through a new paradigm Software Defined Networking, in order to announce the security and efficiency of a network.

Analysis of Network Traffic on Mobile Devices

In recent years, mobile devices (e.g., smartphones and tablets) have met an increasing commercial success and have become a fundamental element of the everyday life for billions of people all around the world. Mobile devices are used not only for traditional communication activities (e.g., voice calls and messages) but also for more advanced tasks made possible by an enormous amount of multi-purpose applications (e.g., finance, gaming, and shopping). As a result, those devices generate a significant network traffic. For this reason, the research community has been investigating security and privacy issues that are related to the network traffic generated by mobile devices, which could be analyzed to obtain information useful for a variety of goals.

Contribution: We survey the state-of-the-art of network traffic analysis of mobile devices. First, we categorize each work according to three criteria: (i) the goal of the analysis; (ii) the point where the network traffic is captured; and (iii) the targeted mobile platforms. Especially, we provide further discussion about the goal related to HCI. To the best of our knowledge, we are the first to survey the works that analyze datasets of mobile traffic that is either: (i) logged on one or more mobile devices; (ii) extracted from wired

network traces; (iii) sniffed at one or more access points of a Wi-Fi network; (iv) eavesdropped by one or more Wi-Fi monitors; (v) produced by one or more mobile device emulators; or (vi) generated via a software simulation. We believe that our classification will help new researchers in this field to foster future research trends about human beings' behavior. This survey is reported in Chapter 5.

Network Traffic Management via SDN

Along with the popularity of the interaction between human and computer, and the heterogeneity of devices, access protocols and current network infrastructure, the management of network traffic and smart devices are becoming more and more vexing, since network administrators perform increasingly sophisticated network management tasks. Moreover, if the network configuration [17] is based on some advanced commands, it is error prone for the users who rely on the Graphical User Interface (GUI) to manage and operate the network. The two main causes of errors are the dynamically changing network state and low-level network configuration. During the last decade, researchers investigated Software Defined Network (SDN) as an approach to aid administrators to manage and program their networks in a more flexible way compared with other traditional network management approaches. SDN functionally separates the network data plane from the control plane. As a result, data plane devices act as a packet forwarding device and leaving the decision making to a centralized system (i.e., control plane). Despite SDN brings many advantages, such paradigm still presents several open issues in terms of security, efficiency, and robustness.

Contribution: We propose an innovative framework named GolfEngine, based on SDN, to simplify the management of network traffic and deployment of security applications to manage the user behavior. GolfEngine enriches the state-of-the-art with the following contributions in terms of components. The first component is "GolfEngine Platform". To the best of our knowledge, we are the first to propose a system-level GUI in SDN management architecture. Users can rely on an intuitive GUI to design their own applications, check on-time network flows, run such applications, and check the network topology in real-time. Moreover, the GUI design of GolfEngine significantly simplifies the management of any abstract filtering rules, minimizing at the same time the risk of network vulnerability due to users' misconfiguration. The second component is "Policy Conflict Detection (PCD)", which guarantees users' security applications to be conflicts free. When a user uploads a new rule to the system, PCD automatically checks policy contradictions and gives a warning to the user if it detects a contradiction. We re-define relations and algorithms for contradictions of two flow rules: $rule1$ and $rule2$. The proposed innovative analysis al-

gorithms to implement discovering conflict assure the robustness, the correctness and the efficiency of GolfEngine. The third component is "Network Statistic Coordinator (NSC)", which is designed for improving the efficiency of communication and lightening the load between controller and switches. The results of our simulation underline that GolfEngine contributes significantly improving the efficiency and security of management HCI impact on the network. The design and evaluation of GolfEngine are presented in Chapter 6.

1.1.3 Energy-Efficient Communication of IoT Devices

The exponential growth smart devices with sensors and networking capabilities have resulted in the development of Internet of Things (IoT) network paradigm. Since an IoT network links smart objects to the Internet, IoT smart devices are having an important role in improving users' experience and life quality. These devices with smart sensors can sense and collect the user data related to Human-Computer Interaction. Some of HCI related devices are battery-equipped and constrained by energy. They interact more with users, they will have more energy consumption. Moreover, in some cases, attackers excessive consuming the energy of energy-constrained devices leads to no more data collected from users. For example, in a scenario, an attacker injects DoS attacks into energy-constrained HCI related devices, which consumes a lot of energy and leads the devices to run out of battery. In order to be resilient against energy attack and efficient in network communication, we analyze the security aspect of an IoT network of battery-equipped HCI related devices from the energy consumption point of view in Chapter 7.

Contribution: We propose EnergIoT, a hierarchical clustering approach based on different duty cycle ratios, which adopts energetic policies to be against energy attack and optimizes the communication among battery-powered HCI related devices. First, we take the energy consumption of idle listening into consideration not only in *network construction* phase but also in *data processing* phase. Second, leveraging on the hierarchical clustering structure, EnergIoT is able to balance the energy consumption in different layers, and thus it avoids *energy hole* problem with assignation of different duty cycle ratios for devices in different layers. Finally, as highlighted by the results of the thorough evaluation we carried out, EnergIoT extends the lifetime of a network of battery-powered IoT devices by 32% more than that of uniform duty cycle approaches without decreasing network performance.

1.2 Publications

Part of the research presented in this thesis and developed during my Ph.D. program produced peer-reviewed workshop, conference and journal publications. The complete list of published and currently submitted works is listed, in chronological order, in Section 1.2.1 (journal papers) and Section 1.2.2 (conference and workshop papers).

1.2.1 Journal Publications

[J1] Riccardo Spolaor, QianQian Li, Merylin Monaro, Mauro Conti, Luciano Gamberini and Giuseppe Sartori. Biometric Authentication Methods on Smartphones: a Survey. In: Psychnology Journal, 14(2-3):87-98, 2016.

[J2] QianQian Li, Sarada Prasad Gochhayat, Mauro Conti and FangAi Liu. EnergIoT: A Solution to Improve Network Lifetime of IoT Devices. In: Pervasive and Mobile Computing, in press, Elsevier, 2017.
DOI: 10.1016/j.pmcj.2017.10.005.

[J3] Mauro Conti, QianQian Li, Alberto Maragno and Riccardo Spolaor. The Dark Side(-channel) of Mobile Devices: a Survey on Network Traffic Analysis. **Under submission at:** Communications Surveys and Tutorials, 2017.
Available as a scientific paper at https://arxiv.org/abs/1708.03766.

[J4] Merylin Monaro, Ciara Galante, Riccardo Spolaor, QianQian Li, Luciano Gamberini, Mauro Conti, and Giuseppe Sartori. Covert Lie Detection Using Keyboard Dynamics. **Under submission at:** Scientific Reprot, 2017. (Minor Revision)

1.2.2 Conference and Workshop Publications

[C1] QianQian Li, Ding Ding and Mauro Conti. Brain-Computer Interface Applications: Security and Privacy Challenges. In Proceedings of the 1st IEEE Workshop on Security and Privacy in Cybermatics (IEEE CNS 2015 workshop: SPiCy 2015), pages 663-666, IEEE, 2015.
DOI: 10.1109/CNS.2015.7346884.

[C2] Merylin Monaro, Riccardo Spolaor, Qianqian Li, Mauro Conti, Luciano Gamberini and Giuseppe Sartori. Type Me the Truth! Detecting Deceitful Users via Keystroke Dynamics. In Proceedings of the 6th International Workshop Cyber Crime (ARES 2017 workshop: IWCC 2017), pages A60(1-6), ACM, 2017.
DOI: 10.1145/3098954.3104047

[C3] Qianqian Li, Reza Mohammadi, Mauro Conti, Chuanhuang Li and Xiaolin Li. GolfEngine: Network Management System for Software Defined Networking. In Proceedings of the 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP 2017), in press, IEEE, 2017.

Part I

Security Issues in User Interaction with Devices

Chapter 2

HCI-based Authentication for Smartphones

According to the participants surveyed in a TIMEs Mobility Poll [18], most of the participants stated that they could not live a single day without their smartphones. In detail, 91% of the participants say that their smartphones are very important and, for 60% of them, even more important than coffee. The pervasiveness of smartphones is generating a symbiotic bound between the user and her smartphone. Lots of researchers put their effort in understanding users behavior using smartphones. Biometric authentication methods move to this direction. Instead of relying on a secret, biometric authentication methods rely on physiological and behavioral characteristics of the user. Moreover, a smartphone is enabled with sensors that are able to infer not only information about the context (e.g., location), but also about its owner [19]. Since smartphones handle a huge amount of private information, recent techniques rely on those sensing capabilities to authenticate the user, measuring her biometric features.

Chapter Organization. The remaining of the chapter is organized as follows. In Section 2.1, we introduce the background for the HCI-based authentication on smartphones. Then we give a clear classification for the HCI-based authentication methods in the literature in Section 2.2. Finally, Section 2.3 discusses some possible future research directions in the field of biometrics authentication methods on smartphones, while while Section 2.4 concludes this chapter.

2.1 Background

Smartphones have become daily used personal devices. People use them for both managing personal data and handling private communications. The presence of such devices is now part of users' everyday life, generating a symbiotic bound between the user and her smartphone. Hence, it is necessary to use authentication methods to prevent the access to un-authorized users. Non transparent methods (e.g., password, PIN) are the most commonly used approaches. However, these methods require an aware interaction by the user and a predefined secret, which can be easily uncover by an attacker. Furthermore, even when in place, these methods often do not prevent a malicious user to get access to the phone (e.g., answering to an incoming call). Moreover, some users tend to avoid these types of authentication, since they are not secure. Indeed, an attacker can uncover and require an explicit user interaction. On the other hand, current transparent methods (e.g., keystroke analysis) take a significant amount of time to authenticate the user, and they cannot guarantee that an unauthorized user is blocked before she gets access to the desired private data or service. For this reason, researchers focused their efforts on designing authentication methods more precise, more usable and less prone to attackers. Biometric authentication methods move to this direction. Instead of relying on a secret, biometric authentication methods rely on physiological and behavioral characteristics of the user.

Smartphones manufacturers (e.g., Samsung) have already expressed their interests in such research topic [20]. Moreover, Biocatch [21] commercializes software products based on cognitive traits (e.g., typical of eye-hand coordination, applicative behavior patterns, usage preferences, device interaction patterns), physiological factors (e.g., left/right handedness, press-size, hand tremor, arm size, and muscle usage) and contextual factors (e.g., device ID, network, geolocation, transaction and navigation behaviors).

In order to protect information on a smartphone from being physically accessed by attackers, a lot of authentication methods have been proposed in recent years. We categorize the literatures by the way how they authenticate users in the following section.

2.2 HCI-based Authentication Methods

Authentication methods are put in place to protect smartphones from being accessed by un-authorized users. Common authentication methods that are already deployed on smartphones leverage a secret information which should be known only by the authorized user. Such methods are passwords, Personal Identification Numbers (PIN) and unlock patterns. In recent years, researchers proposed a lot of authentication methods. We categorize the

HCI-based biometric authentication methods in the literature by the way they perform the user authentication.

- Firstly, it is possible to categorize the authentication methods according to the nature of the biometric features that are measured by smartphone sensors (schematized in Figure 2.1):
  - **Physiological biometrics:** measure user's body characteristics (e.g., fingerprint, face).
  - **Behavioral biometrics:** are metrics that build patterns according to the way the user behaves with her mobile device (e.g., keystrokes dynamics, gait).
- Secondly, biometrics authentication methods can also be divided into two main categories according to time required to collect data from users and authenticate them:
  - **One-time authentication:** requires the user to perform a specific task for a limited period of time (typically few seconds). In order to not affect the user experience, an one-time authentication method needs to be as fast as possible, but at the same time it must reach an high accuracy. For the same reason, such authentication is performed only on privacy sensitive phases (e.g., smartphone unlock, access to bank account).
  - **Continuous authentication:** involves the collection of data from sensors for a long period of time. Practically, a continuous authentication method performs two tasks at the same time: (i) incrementally builds a behavioral profile of the user adding new observations from sensors, (ii) verifies that the current observation matches with the behavioral profile of the user built from the past observations.

It is worth of notice that, at the best of our knowledge, all physiological biometric-based authentication methods in the literature belong to one-time authentication category.

Another distinction that could be done is the one between *implicit* (or transparent) and *explicit* (or non transparent) authentication methods. On one hand, explicit authentication methods require the user to perform a specific task, so the user is aware of when the authentication method is taking place. On the other hand, implicit ones do not require the user to perform any specific task to be authenticated.

2.2.1 Physiological Biometrics

Several authentication methods that rely on physiological biometrics are already deployed on smartphones. As a first example of physiological bio-

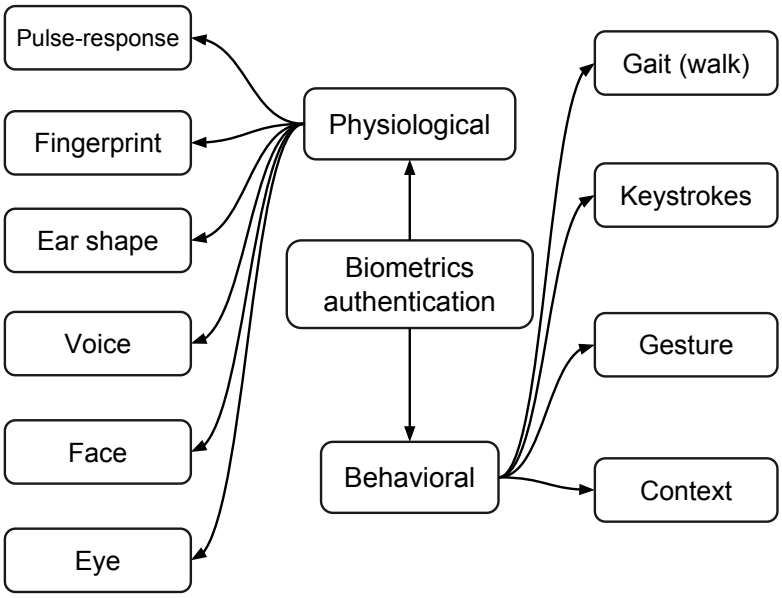


Figure 2.1: Biometrics authentication on smartphones.

metric, manufacturers recently started to embed on high-end smartphones a specific biometric sensor for digital fingerprints. As another example, authentication systems use the frontal camera in order to recognize the face or the eyes of the owner. Hadid et al. in [22] propose a face and eye detection and authentication for mobile phones. For the face detection, there are two approaches: color based (fast but prone to illumination and background changes) and Haar-like/AdaBoost based (slow but accurate). The first is color-based face detection that is simpler, faster without robustness against illumination and background changes. The second is Haar-like/AdaBoost based Face Detection. The Haar-like features are extracted using the notion of integral image, while AdaBoost is used to select the most prominent features among a large number of extracted features and construct a strong classifier from boosting a set of weak classifiers. This method proves to be slower, but more effective and accurate. In [23], the authors propose a standalone modular biometric system according to periocular information to authenticate users on smartphones. In the first stage, this method makes users to capture a probe periocular image with rear or front camera and stores into the database for extracting features. The extracted features are compared against all reference templates in the database, to obtain a comparison score which decides upon the access to the device. Ali Fahmi et al. in [24] propose an authentication method that relies on the uniqueness of the shape of human ear. Using the frontal camera, the method is able to authenticate the user by the shape of her ear while she is taking a call. De-

spite the prototype was not originally deployed on smartphones but it can be extended also to them, Rasmussen et al. in [25] propose a pulse-response biometric authentication method. Pulse-response biometric relies on the property that each human body exhibits a unique response to a signal pulse applied at the palm of one hand, and measured at the palm of the other.

Other researchers rely on a set of physiological biometrics at the same time. Kim et al. in [26] propose an enhanced authentication method using multi-modal personal information. The proposed approach collects information from face, teeth and voice from smartphone sensors, and authenticates users using these three traits simultaneously. Moreover, Boehm et al. in [27] proposed a system for secure smartphones login with authentication combining face recognition with gaze tracking. During the login, when a face recognizer continuously checks the identity of the user, a number of icons are displayed on the screen; as the icons move in incongruent line patterns, the user follows her secret icon with her eyes.

2.2.2 Behavioral Biometrics

In this section, we report the most relevant authentication methods that use behavioral biometrics. Some of them are not strictly designed for smartphones but they can be extended to work also on such mobile devices. We first describe one-time authentication methods, then we proceed with continuous time ones.

One-time authentication methods

In the literature, some researchers rely on behavior of the user while she inputs information on touchscreen. Giuffrida et al. in [28] propose a sensor-enhanced keystroke dynamics authentication method. In other words, while users input the password, the system authenticates them using both motion sensors and taps on the touchscreen. The effectiveness of this authentication method is also proved as secure even against the statistical attacks [29]. The authentication method proposed by Zheng et al. in [30] is similar to [28]. However, the authors in [30] combine four features (acceleration, pressure, size, and time) extracted from smartphone sensors as verification mechanism to authenticate whether the user is the true owner of the smartphone or an impostor who happens to know the password. Another method proposed in [31] performs an implicit authentication where users are authenticated by the way they perform the unlock pattern on touchscreen. De Luca et al. in [9] proposed a method where users draw shapes (or gestures) on the front and the back of smartphones to enter tap-based passwords. The sequence and the way users draw the shape password is one kind of authentication which is a good resistance against shoulder surfing attacks and is easy and fast to use. This method also uses the habit they switch sides of the smart-

phones as authentication which increases security while authentication speed stays relatively fast. Another example of authentication with touchscreen is proposed by Saevanee et al. in [32], where the authors use the finger pressure on a touchscreen to authenticate the user. Other work in this direction uses motion sensors to authenticate implicitly the user when she is answering or placing a phone call [33], also called phone-to-ear gesture.

Continuous authentication methods

A behavioral biometric metric is user's gait (i.e., the way she walks). The authors in [34] test subjects' gait while they wear an accelerometer device on their belt. This method identity users with portable devices while they walk with devices. Similarly, the method in [35] assumes that the smartphone is placed at the hip of each volunteer to collect gait data. Regarding the user interaction with the touchscreen, Frank et al. in [36] propose a continuous authentication method based on user interaction with smartphones' touch-screen (i.e., updown and leftright scrolling). Similarly to the one-time method [28], Gascon et al. in [37] propose a continuous authentication method that analyzes typing-motion behavior of users on smartphones. The authors recruited participants to let them enter some short sentences on a touchscreen while all available sensor events were recorded to build a typing-motion behavior profile of the user. Some other methods consider the context and location in which a mobile device is used. Shi et al. in [38] present an implicit authentication, an approach that uses observations of user behavior for authentication. The authors collect users behavior data for two weeks when they use their smartphones as authenticating method. These devices can collect a rich set of information, such as location, motion, communication, and usage of applications. This implicit proves to be more suited for mobile devices. The authors collected users' behavior data from their smartphones for two weeks in order to prove to the effectiveness of their proposal. Another example of context inferring technique is CRêPE [39], a framework for enforcing fine-grained context-related policies, which can recognize the context in which a mobile device is used, continuously monitoring the environment via phone sensors. This framework supports both physical contexts (i.e., location, time), which are associated with physical sensors (i.e., GPS, clock, Bluetooth, etc.), and logical contexts, which are defined by functions over physical sensors. The authors in [40] proposed a framework to analyze encrypted network traffic and to infer which particular actions the user executed on some apps installed on his mobile-phone. Conti et al. in [33] investigated a new biometric measure to authenticate the user of a smartphone: the movements the user performs when answering (or placing) a phone call. The biometric measure leverages features that are becoming commodities in new smartphones (i.e. accelerometer and orientation sensors).

2.3 Future Research Directions

Some possible future directions on continuous authentication methods can rely on cognitive games or other sources of information such as external devices and side-channels.

2.3.1 Cognitive Games

A possible future direction in one-time behavioral authentication consists of measuring the behavior of the user while she solves cognitive games. Existing cognitive game easily extendable for this purpose are cognitive CAPTCHAs for touchscreen-enable devices, such as Capy Puzzle [41] and CAPTCHaStar [42].

2.3.2 External Devices

An authentication method relies on an external devices to authenticate the owner of a smartphone. Additional sources of information to improve the reliability of existing authentication methods could be external or wearable devices, such as smartwatches, fitness wristbands or Google Glass devices [43]. In fact, such devices are embedded with sensors, thus they could be a source of valuable information to infer behavioral patterns. In this work, we do not consider other physiological biometrics such as blood and DNA. This because they require specific hardware that are not commercially available at the current time, but it is reasonable to consider that such hardware could be embedded into smartphones in a near future.

2.3.3 Side-channel Analysis

A side-channel is an observable source of information that is the result of the way the user interacts with a device. Network traffic is an example of side-channel and it could be relied on to build a behavioral profile of a user. Some recent work shows that it is possible to infer from the encrypted network traffic the set of apps installed [44] and even the actions the user performs within an app [45]. Moreover, since Conti et al. prove that it is possible to recognize a user from the energy consumption of her laptop [46], it is reasonable to consider this side-channel on smartphones as a valuable source of information for a user authentication method.

We strongly believe that it is possible to build an authentication method combining multiple behavioral biometrics and evaluating their impact on authentication dynamically, with an approach based on the context of usage. We also believe that, in the future, biometric authentication methods will significantly improve both the security and the usability of smartphones.

2.4 Summary

In this chapter, we surveyed the state-of-the-art of authentication methods on smartphones that are based on user biometrics. Firstly, we categorized authentication methods in the literature according to the nature of the biometric features, which are measured by smartphone sensors, into behavioral and physiological biometric categories. Physiological biometrics are related to body characteristics of the user, while behavioral biometrics are related to the way a user interacts with her smartphone. Secondly, we divided the behavioral authentication methods into two categories according to time required to collect data from users and authenticate them (i.e., one time and continuous authentication). Finally, we discussed some possible future research directions in the field of biometrics authentication methods on smartphones.

Chapter 3

HCI-based Security Challenges in Brain-Computer Interfaces

Brain computer Interface (BCI) is a communication system between the brain and the external environment. It collects data related to users' brain activities through sensors and transfer this data to computers. In BCI systems, the brain does not use peripheral nerves in order to give orders to our body. Instead, the orders are captured directly by BCI devices and encoded into electro-physiological signals. These signals become commands that can control external devices and computer applications. For example, in order to control a cursor, signals are transmitted directly from the brain to the application that moves the cursor, rather than taking the "route" through peripheral nerves from the brain to the hand to move a mouse. This technology makes it easier for a human to communicate with computers or external devices, such as prosthetic devices (especially for the patients with severe neuromuscular disorders). With the development of intelligentization, BCI technology has been pervasive in several fields of our life, such as, neuromedical field, authentication, gaming, entertainment, and marketing. Unfortunately, BCI manufacturers are developing devices and applications without taking much the security and privacy issues into account. Using such devices, individuals' private information could be stolen by malicious third party applications. Hence, it is necessary to discover the security issues on this new technology and make it better sever the authentication function.

Chapter Organization. The remainder of the chapter is organized as follows. Section 3.1 describes background knowledge of Brain-Computer Interface. In Section 3.2, we revise current state-of-the-art BCI applications (i.e., neuromedical applications, authentication, gaming and entertainment, and smartphone-based applications). Then in the following Section 3.3, for

each application scenario we discuss the key security and privacy challenges together with possible countermeasures. Finally, we make a summary of this chapter in Section 3.4.

3.1 Background

In this section, we give background information of the working principle of BCI, the classification of BCI system according to the security level, and the BCI devices comparison. Figure 3.1 shows the working of brain-computer interfaces. First, the brain neural signals are captured by BCI devices (step 1): this process is named signal acquisition. After signal acquisition, BCI systems transform these analog signals into digital signals (step 2). Then, using signal processing, the features are extracted and classified (step 3 and step 4). Then, the signal output is sent to BCI applications (step 5).

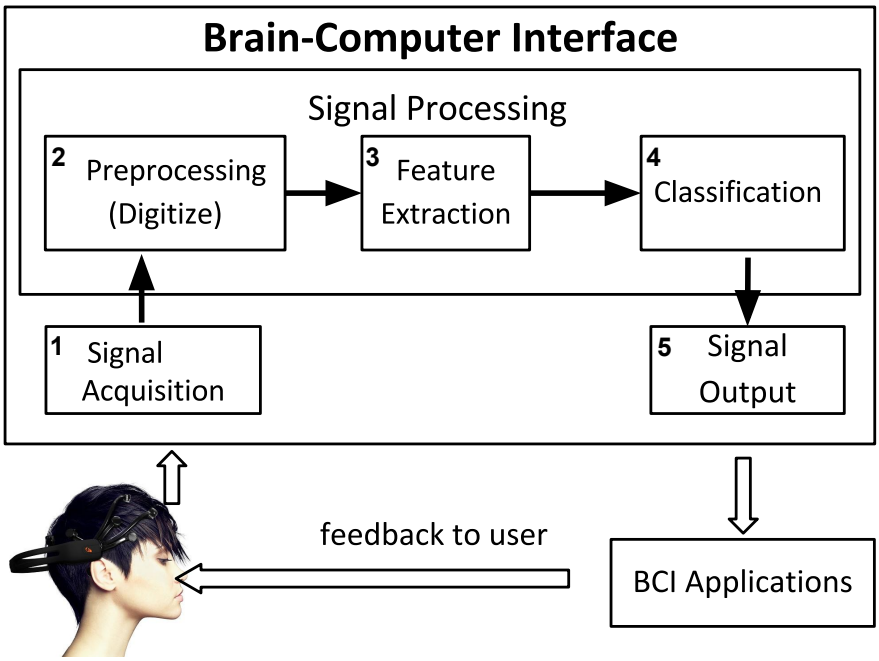


Figure 3.1: The work principle of BCI.

The brain has always fascinated human beings, and particularly a German scientist named Hans Berger, who discover electroencephalography (EEG) [47] about 80 years ago. After this, new methods for exploring it have been found and we can categorize them into three main groups: invasive, partial invasive and non-invasive. BCI systems could be classified into three main groups: 1) invasive system [48], 2) partial invasive system, and

3) non-invasive system. An invasive system requires physical implants of electrodes into the grey matter of the brain by neurosurgery, which makes it possible to measure single neurons or local field potentials. Partially invasive system is another brain signal reading process which is applied to the inside the skull but outside the grey matter, e.g., electrocorticography (ECoG). It records the activity of the brain inside the skull, but on the surface of the membranes that protects the brain. A non-invasive system (e.g., electroencephalography - EEG, and functional Magnetic Resonance Imaging - fMRI) is the most frequently used neuron signal capturing method. This system is applied to outside of the skull, just applied on the scalp. It records the brain activities inside of the skull, and on the surface of the brain membranes. Both EEG and fMRI give different perspectives and enable us to "look" inside of the brain [49]. In EEG, brain-related electrical potentials are recorded from the scalp. Pairs of conductive electrodes, are used to read electricity. Note that, invasive and partial invasive systems are prone to scar tissue, and they are difficult to operate. Furthermore, both of them are quite expensive. Although EEG signals can be effected by noise and signal distortion, they are easily measured and have good temporal resolution. Therefore, the most widely used method for recording brain activity in BCI systems is EEG. EEG-based devices directly measure electrical potentials produced by brain's neural synaptic activities. Five waves from human brain activities that could be captured by EEG devices are shown in Figure 3.1.

Table 3.1: Brain waves.

Type	**Frequency (Hz)**	**Normally**
Delta	0.5 - 4 Hz	hypnoidal
Theta	4 - 7 Hz	slow wave sleep
Alpha	8 - 12 Hz	stable wave
Beta	12 - 30 Hz	action wave
Gamma	Over 30 Hz	arousal and excitement

Event Related Potential (ERP) is measured by means of EEG. An ERP is detected as a pattern of voltage change after a subject responses to a certain stimulus. Every ERP is time-locked to the stimulus, i.e., the time frame at which an EEG voltage change is expected to occur is known given the timing of the stimuli. The study of the brain in this way provides a noninvasive means of evaluating brain functioning in patients with cognitive diseases. Exposure to external stimuli, such as a tone or light flash, can generate responses in the EEG wave. Internal stimuli, like skipping an expected stimulus can also generate a response. In both cases this is called Event Related Potentials (ERP). Currently several companies produce BCI devices, for different purposes, ranging from clinical-grade BCI devices to

consumer-grade BCI devices. Table 3.2 lists the main features of three common devices, while Figure 3.1 shows the appearances of these devices.

Table 3.2: Comparison of BCI devices.

Device	**Price**	**Electrodes**	**Resolution**	**Interface**
BioSemi Active [50]	$12000	256	24 bits	Wired
Emotiv EPOC [51]	$399-499	14	14 bits	Wireless
NeuroSky [52]	$50-150	1	8 bits	Wireless

Figure 3.2: BCI devices.

3.2 BCI Applications

Brain-computer interface was first developed for assistance, especially for the patients with severe neuromuscular disorders. More recently, however, BCI have had a surge in popularity for non-medical uses, such as gaming, entertainment and marketing. In this section, we classify BCI applications into four different application scenarios according to their usage purpose (i.e., neuromedical applications, user authentication, gaming and entertainment, and smartphone-based applications). For each application scenario, we provide a description.

3.2.1 Neuromedical Applications

Since BCI technology makes it easier for a human to communicate with external devices or computers, it is widely used in the neuromedical area to help patients to control their body through BCI devices instead of nerves. BCI technology can help patients, especially with serve neurological disorders, e.g., Parkinson disease. Several neural implantable devices [53] will be available in the near future. Because of being closely related with health, security and privacy concerns become especially necessary to be taken into consideration. An example of neuromedical applications that might be exposed to attacks is prosthetic limb application [54].

3.2.2 User Authentication

Authentication is a process that ensures and confirms a user's identity. It plays an important role in security systems. Technology advancing makes it possible to easily obtain EEG signals. Using EEG brain signals as authentication measure has been proposed in many literatures and proved to be effective. Authors in [55] aim at authenticating users, based on brainwave signals. In particular, they use single-channel EEG signals to do authentication. In this authentication system, BCI devices record brainwave signals when a subject performs a custom task (e.g., singing, breathing or finger movement). Then, brain signals are wirelessly transmitted to a computer application which collect and process this data. Their authentication system analyses the similarity between such brain data and training data to authenticate subjects. The authors show that their proposed authentication mechanism has the same accuracy as multi-channel EEG authentication, about 99% accuracy. Similar to [55], authors in [56] take EEG brainwave features as neural passwords to do authentication. The entire process is performed automatically, without human supervision. The authors use an algorithm that automatically extracts neural events corresponding to an individual's blinking, jaw-clenching, and eye-rolling activities. The results show that accuracy of this authentication method ranges from 67% to 95% with single-trial inputs.

3.2.3 Gaming and Entertainment

With the development of BCI technology, there are several BCI games available in entertainment industry [57] [58] [59]. The principle of most BCI games works in a way similar to P300-speller. In this kind of games, an amplitude peak in the EEG signal is detected at more or less 300ms after a stimulus. In this kind of games, an amplitude peak in an EEG signal is be detected at more or less 300ms after a stimulus. In the game P300-speller, stimuli are alphanumeric characters shown on the screen. The characters are arranged in a matrix where rows and columns flash on a screen in a rapid

succession. According to the being spelled word, users choose one character using eyes from the screen. Through analyzing peaks occurring in the brain-waves, authors get the spelled word. In [60] the authors analyze the P300 BCI features in gaming applications and show that the P300 BCI even with some shortcomings, not only exhibits relatively high speed and accuracy but also can be used without user training, after a short calibration. In addition, we list four classic P300-based BCI games here.

- MindGame [59]: In this game, the user moves a character from one field to another on a game board. The length of the characters steps depends on the BCI classifier output, with a stronger brain response to target flashes leading to larger steps and the faster attainment of the game goal, which is to visit all predetermined target fields. Unlike of typical P300 BCI spellers, the stimuli are not presented group-wise (as in rows and columns) but rather separately, one at a time ("single-cell" or "single-character" design).

- Snake [61]: It is also based on the same principle of P300-speller. In this game, a snake can move in three directions: forward, left and right. The goal is to locate and eat apples on a map. Eating apples makes the snake grow in length, and becomes as large as possible. For the sake of having speed in the game, moving forward is automatic, and both turning left or right is controlled by the user via EEG signals.

3.2.4 Smartphone-based Applications

The application scenario we want to consider in this subsection is actually mainly driven by a specific emerging and pervasive technology, i.e., smartphones. Along with the advances in smartphone capabilities, there is an increasing interest in using smartphone by individuals in their daily life. BCI are used in conjunction with this technology (smartphone). Recently, some BCI applications based on smartphone have been proposed in many literatures. Authors in [62] implement a brain-controlled address book dialing app, which works in a way similar to P300-speller. Instead of showing characters in P300-speller, the dialing app shows a sequence of photos of contacts from the address book. Therefore, the user can easily select a person whom she or he wishes to dial. In this application, EEG signals from the headset are transmitted wirelessly to an iPhone, which natively runs a lightweight classifier to discriminate P300 signals from noise. When a persons contact-photo triggers a P300, his/her phone number is automatically dialed. Authors in [63] measure a subject's attention and meditation level through EEG signals when a subject is playing a game. Authors compare the difference among all the subjects' EEG signals, according to subjects' age and gender. Their results show that, in the POKOPANG game, the average attention level of men is lower than that of women, while the meditation

level is reversed. As a result, authors infer that women are more interested in POKOPANG game. Air Brain system [64] is a portable EEG telemetry system. Different from other portable EEG monitoring systems, in order to have more storage space, this way, the stored data can be accessed from everywhere. To achieve this, the system uses 3G network of smartphone to transfer data. Air Brain system enables subjects to measure EEG signals immediately after subjects start walking. Furthermore, the system is able to detect eye closing by measuring changes of aplha wave.

3.3 BCI Security and Privacy Challenges

In this section, we describe possible attacks (either already doable, or envisaged to be possible in the near future) for the above scenario. Finally, for each application scenario we also discuss possible countermeasures.

3.3.1 Neuromedical Applications

BCI devices in medical care industry could help patient with neurological disorders, such as injury, spina bifida, stroke, encephalitis, multiple sclerosis, Parkinson disease, and Alzheimer disease. However, there exist some attacks [53]. As a representative case of neuromedical applications, prosthetic limb application [54] and Deep Brain Stimulator [65] are two classic applications. In what follows, we list possible attacks and countermeasures for Neuromedical applications scenario.

Prosthetic Limbs

- **Attack Model:** Prosthetic limb application [54] allows physicians to connect wirelessly to adjust settings of neural implant devices. If complete brain neural signals are transmitted, an attack can intercept the transmission, save brain neural signals, and decompose the raw signals to obtain private information. An attacker could try to hijack these signals to take control of the robotic limb or give dangerous movement feedback to the patient. We underline that these attacks are possible even when information is transmitted in an encrypted format [40]. Furthermore, attacker could try to control prosthetic limbs of patients and give dangerous movement to patients. Under this condition, the attacker does not need to be near the patient but only needs to have attack hardware placed near the patient. The attacker could also infect the patient biotechnology components with a digital virus. Another possible scenario is the case in which patients are attackers who might modify settings on their own prosthetic limbs. They might just want to override mechanical safety settings to gain

extra strength or interfere with limb feedback to eliminate the ability of feeling pain.

- **Countermeasure:** There are some appropriate safeguards in the design of the prosthetic applications which can be deployed in the coming years. For these prosthetic applications used to give treatment for patients, it is clear that the main countermeasures should focus on preventing life-threatening attacks. Also, we should protect private feelings and emotions of patients from being leaked to attackers. For these prosthetic systems, it is necessary to guard against attackers trying to prevent the patient from using the limb, particularly since this can occur while the user is running, driving. In addition, these applications should prevent attackers from remotely eavesdropping on the wireless signals and collecting private information about patients' activities. The communication between neural implantable devices and patients must be kept confidential. Furthermore, if they are in sensitive condition such as depression, trying to prevent wireless attackers from detecting the presence of these implant devices is effective to protect safety. In the future, more effective countermeasures should be proposed to guarantee that prosthetic applications are not only safe and effective, but also these applications are robust enough to prevent attacks.

Deep Brain Stimulator

- **Attack Model:** The Deep Brain Stimulator System (DBSs) is a vibration tool that will assist medical doctors in determining the implementation site for the electrodes of a deep brain stimulator. Current-generation DBSs has had success treating Parkinson disease, chronic pain, and other medical conditions [66] [67] [68]. But these devices are vulnerable to the attack. For example, patients may attempt to modify setting to make themselves' moods feel good, whereas attackers may also attempt to program the stimulation therapy maliciously. The attacker's strategy does not need to be too complexed if they want to cause harm. It can also even cause cell death or the formation of meaningless neural pathways by bombarding the brain with random signals. Alternatively, an attacker might wirelessly prevent the device from operating as we want.

- **Countermeasure:** We must also ensure that DBSs protect the feelings and emotions of patients from external observation. Furthermore if he or she is receiving treatment for a socially sensitive condition such as depression a patient might also want to prevent a wireless attacker from detecting the presence of the DBS.

In the future, more effective countermeasures should be proposed to guarantee that neuromedical applications are not only safe and effective, but also robust enough to prevent attacks.

3.3.2 User Authentication

Using EEG brainwaves to authenticate might result in risks for the privacy of users.

- **Attacks:** Using EEG brainwaves to authenticate might result in risks for the privacy of users. For example, authors in [69] propose an authentication system that verifies an individual EEG signal when a subject performs a custom task (e.g., singing, breathing or finger movement). They also design an attack model by impersonating the thoughts of subjects. The authors make deliberate attacks from thought impersonators to test the robustness of the authentication system. Similar to [70], an adversary can attack the authentication system via synthetic EEG signals, using EEG generative model based on the historical EEG data from a subject can also attack the authentication system [71].

- **Countermeasures:** To mitigate the authentication attacks mentioned about, a possible way is to reduce authentication error rate. For example, we can enlarge the number of participants, use recruited attackers, and integrate the data processing methodology with a real-time authentication framework to achieve reduced authentication error rate. The method in [69] is like the authentication of fingerprint. In order to enhance this system robustness, the authors should increase the number of test subjects, use recruited attackers, and integrate the data processing methodology with a real-time authentication framework. Moreover, the system maintedevelop data processing enhancements that further reduce the authentication error rates in general. Chuang and John in [72] propose one-step two-factor authentication. Two-step verification does not mean two-factor authentication and two-step verification. These two factors are the knowledge factor and the inherent factor. For example, reading the password is a well-established behavioral biometric that fits the one-step two-factor criteria. In this method, when a user read a password (the knowledge factor), the signals when reading (the inherent factor) is also employed to authenticate the user. Therefore, if using some more secret knowledge, the authentication system will be more effective against attackers. Moreover, another possible method to enhance the robustness of authentication is by leveraging multidimensional method [73]. For example, using multiple authentication signals (e.g., the signals of singing, breathing,

or being shocked). Besides, we can combine the existing authentication methods on smartphone device to perform multidimensional authentication.

3.3.3 Gaming and Entertainment

Brain-computer interfaces are becoming increasingly popular in the gaming and entertainment industries. Martinovic et al. [74] highlight the existence of side-channel attacks by malicious third-party games on BCI devices.

- **Attacks:** Similar to smartphone games, third-party BCI games depend on common APIs to access BCI devices. Thus, such APIs supply unrestricted access to raw EEG signals for BCI games. Furthermore, such games have complete control over the stimuli that can be presented to users. As a consequence, attackers can display the contents and read their corresponding EEG signals. The content might be videos, pictures, or numbers, which users see when they playing games. Therefore, attackers can specifically design some videos and images shown to users in order to maximize the amount of leaked information. In particular, the impact of exploiting or mishandling BCI devices is difficult to estimate. Authors in [74] demonstrate BCI games could be exploited to extract individuals' private information, such as 4-digit PINs, bank information, date of birth and location of residence, using users' recorded EEG signals.

- **Countermeasures:** Authors in [75] identify security and privacy issues arising from possible misuse or inappropriate use of "Brain Malware" information. In particular, they propose an interdisciplinary approach to enhance the security of BCI systems by the aid of several experts from different areas, such as neuroscientists, neural engineers, ethicists, as well as legal, security and privacy experts. Authors in [76] propose a tool named "BCI Anonymizer" to prevent the side-channel extraction of users' private information. The basic idea of the "BCI Anonymizer" is to remove private information from raw EEG signals before this information is stored and transmitted. "BCI Anonymizer" could be implemented either in hardware or in software, as a part of BCI devices, but not as part of any external network or computational platform. Moreover, the "BCI Anonymizer" can generate an anonymized neural signals to replace the removed signals that represent private information. However, authors in [76] do not provide a clear method to distinguish the difference between users' private information and commands to applications.

3.3.4 Smartphone-based Applications

The smartphone-based BCI applications are prone to attacks that originate in the mobile device itself. Therefore, most of the possible attacks on smartphone issues could also be considered as security and privacy issues of smartphone-based BCI applications. These attacks could be considered as internal and external attacks.

- **Internal Attacks:** Smartphone applications can access private data which is acquired from BCI devices and stored in smartphones or SD card. This data can be illegally transferred by a malware to a remote server (e.g., privilege escalation attacks [77]). Developers of malwares can analyse the private signals and attack the users of BCI devices. Attacks to smartphone applications also apply to smartphone-based BCI applications. Another attack is under the condition of FM Radio Data System (RDS), where the data channel connects all devices (especially with smartphones) with FM radio receiver chips. Despite the potential risk, RDS has been so far completely ignored by security providers, as discussed in details in previous work [78].

- **External Attacks:** the potential attack could be from outside of smartphones. As a presentative case, here we consider the FM Radio Data System, RDS. Open and broadcast in nature, this data channel connects all devices with FM radio receiver chips, which are common for Android devices commercially available. Despite the potential risk, RDS has been so far completely ignored by security providers, as discussed in details in previous work [78]. Another kind of external threats is based on the Man-In-The-Middle principle. When users access the Internet by the public access point, the improper use of the SSL protocol might cause attacks [79].

- **Countermeasure:** Given that we are considering BCI applications in conjunction with a specific technology (smartphone), here countermeasures are mostly the ones typical for generic smartphone security. Useful security approaches could be the ones that track the flow of information. For example, TaintDroid [80] proposes a model that can track not only the way applications access sensitive data, but also the way applications use such data. FlowDroid [81] proposes an innovative and accurate static taint analysis for applications in Android platform, allowing proper analysis to handle callbacks invoked by the Android framework. In addition to the aforementioned approaches, fine-grained context-based access control [39] is another effective way to limit the leakage of private data. These mitigations are possible only by modifying Android's permission model, e.g., Android's internal middleware layer. Moreover, we can modify the Android system

permission model to prevent the external threats. To achieve this, the middleware layer of Android architecture is needed to recode. Another method is to prevent the external threats at application level, which is easily deployed.

3.4 Summary

In this chapter, we review main common brain-computer interfaces (BCI) applications, and their possible security and privacy issues. Moreover, we consider four different application scenarios: 1) neuromedical applications, 2) user authentication, 3) gaming and entertainment, and 4) smartphone-based applications. For each scenario we provide the description of current state-of-the-art technologies, potential attacks that might threat each scenario, and envisaged countermeasures.

Chapter 4

Detecting Deceitful Users via Keystroke Dynamics

In recent decades, interacting remotely with online services is becoming extremely common for the users. However, in these interactions, users provide personal and private information for some ecommerce services, especially via keyboard. For example, while online banking is a very convenient way to access bank services, fraudster that gained personal data has an easy access to the bank account of the cheated person. Besides, the increasing popularity of Online Social Networking (OSN), such as Facebook, Twitter and LinkedIn, has been accompanied with an increasing interest in the risk to be attacked and manipulated. Since OSN services are open to all the Internet users, they are particularly subjected to Sybil attacks [82], during which a malicious user creates one or multiple fake OSN profiles. These kinds of frauds are known also as Fake Profile Attacks (FPA) [83]. In FPA, real personal information is used to create a fake profile that takes the identity of an unsuspecting user. There is a large number of reasons why people create fake accounts on OSN. We can classify them into two categories: social reasons and business motivations. Social reasons include, for instance, friendly pranks, online gaming, stalking, cyberbullying, and concealing real identity to bypass some real-life constraints [84]. These cheating may cause a lot of problems in our daily life, such as in stalking cases and other privacy issues, but they have less impact on the online community in terms of cybersecurity. On the other hand, business motivations lead to use fake accounts for more malicious behaviors in the online security context. In all these scenarios, the truthfulness of the information provided by remote users becomes a key issue, mostly concerning the user authentication procedures. It has been reported that, on February 2010, 1.5 million Facebook accounts were on sale on the black market [83, 85]. This is the reason why Facebook

is called with the pseudonym of Fakebook. Fake accounts are bought or created to increase the visibility of niche content, forum posts, and fan pages by manipulating votes or visit counts [86]. Spammers abuse of OSN messaging systems to post junk emails or advertisement, inducing customers to pay for clicks. Furthermore, fake accounts can be used to access personal user information and perform large-scale crawls over social graphs. During the registration on a OSN, if the system is able to detect when a user provides false personal information, the creation of fake profiles will become harder, significantly reducing the feasibility of Fake Profile Attacks (FPA). Given these issues and the vast amount of personal sensitive data shared on online services (e.g., living location, Pin-Code), scientific literature proposed different measures to prevent accounts being attacked and cloned [87], but the problem has not yet been solved effectively.

The aim of this chapter is to propose and evaluate a new tool able to detect deceptive users through the interaction between users and keyboard. More precisely, we want to clarify that using keystroke analysis makes it possible to detect people providing false personal information during the authentication on an online service.

Chapter Organization. The remainder of the chapter is organized as follows. In Section 4.1, we review current state-of-the-art related to keystroke dynamics. Then in the following Section 4.2, we give a detail illustration of our tool. We report our results in Section 4.4. Finally, we make a discussion and summary of this chapter in Section 4.5 and 4.6, respectively.

4.1 Related Work

In this section, we first report some background concepts on the functioning of human cognitive system during the processing of a deceptive information. Then, we survey the work on keystroke dynamics as a way to detect deception.

4.1.1 Cognitive Basis of Deception

Cognition is largely involved in the process of lie [88]. One of the main evidences relates to the increase in cognitive load associated with falsehood productions [89]. Generally, a greater cognitive load produces a bad performance in the task the subject is carrying out, in terms of timing and errors [90]. In particular, subjects manifest a lengthening of reaction times (RTs) and an increase in errors rate. Since lying is more cognitive demanding than telling the truth, our cognitive system becomes slower when we have to produce a lie [91]. According to the literature, this is probably due to the fact that in a falsehood condition we don't simply have to produce an answer, but we have to inhibit the "true" answer and, subsequently,

produce a "false" answer [92]. The generation of a lie requires to monitor the reaction of the interlocutor and to adjust the behavior congruently to the lie [93]. This phenomenon can be observed by studying the RTs in a double choice task: the choice between two alternatives becomes slower in the deceptive responses than the truthful responses [94]. According to the functioning of our cognitive system, behavioral-based lie detection tools have been proposed to identify liars, such as the RT-based Concealed Information Test (RT-CIT) [95] and the autobiographical Implicit Association Test (aIAT) [96]. CIT and aIAT have been shown to have a lie detection accuracy around 90% and are generally applied in legal context, for example to determine which one between two alternative versions of a crime is true. More recently, Monaro and colleagues developed a novel technique [97] that is able to identify liars about identity based on the mouse dynamics analysis with an accuracy around 95% [98, 99, 100]. The technique requires the suspect to undergo a computerized test in which unexpected questions about the claimed identity are presented on a computer screen. The subject is required to respond using the mouse: the trajectory is recorded in its spatial-temporal characteristics, and then classified by machine learning methods as coming from a truthful or deceptive responder. This is the first technique in literature that exploits human-computer interaction to spot faked autobiographical information, as identity. Indeed, it is the first lie detection tool that allows to establish if an information is true or false, unlike CIT and aIAT that need that two alternatives are given to identify the true and the false one. Since RTs and mouse dynamics are considered reliable behavioral indices of deception, keystroke characteristics may provide a clue for recognizing deceits. The authors in [14], t proposed a Keystroke Dynamics Deception Detection model to explain the relationship between deceptive behavior and keystroke dynamics. According to this model, the production of a falsehood may cause an increase in both emotional arousal and cognitive load. These may result in a consequent change in the fine motor control, which in turn results in a deviation of the typing ability, affecting the keystroke dynamics personal baseline.

4.1.2 Deception Detection via Keystroke Dynamics

Keystroke dynamics is the detailed timing information about human typing rhythm: it describes exactly when each key is pressed and released, while a person is typing at a computer keyboard, a mobile phone or a touch screen panel [101]. The typing pattern analysis could be considered as an implicit behavior measure, because users are not aware of it during the interaction with the device [102]. Furthermore, keystroke dynamics is unique and characteristic for each individual [103]. For this reason, keystroke dynamics has been used for security purposes, such as user authentication [104] and user identification [105], to differentiate between two or more different users. Re-

searchers have also demonstrated that it is possible to investigate changes in typing behavior within subjects, for example in emotion discrimination [106] or in the distinction between stress and non-stress conditions [107].

According to literature, some studies focused on extracting deception from typed text, but the majority of these used a linguistic approach or considered only some simple features of the text rather than the rhythm of typing [108, 109]. Mihalcea et al. in [108] collected deceptive and truthful personal statements about given topics from Amazon Mechanical Turk participants. Using a classifier based on psycholinguistic analysis, they obtained an accuracy of 70% in classifying true and deceptive statements. Ott et al. [109] also used Mechanical Turk to generate a dataset of 400 deceptive hotel reviews. These were combined with 400 truthful reviews from TripAdvisor about the same hotels. True and false reviews were used to train a machine learning classifier based on three different approaches (genre identification, psycholinguistic analysis and text categorization) that could distinguish deceptive from truthful reviews at the 90% accuracy level. Zhou et al. in [110, 111] investigated if behavioral indicators can be used for deception detection in instant messaging. They explored different nonverbal and verbal behaviors during a chat discussion between participants, showing that these indices could significantly differentiate deceivers from truth tellers, such as participation level, discussion initiation, cognitive complexity and non-immediacy of sentences, frequency of spontaneous corrections, lexical and content diversity. Derrick et al. submitted the participants to a computer-mediated chat-based, instructing them to be deceitful or truthful in response to questions, according to a prompt given by the system. The Chatterbot captured four main indices: response time, number of edits (basic keystrokes as backspace and delete), word count and lexical diversity. The results support cognitive load theory, confirming that deception is positively correlated with the response time and the number of edits, but negatively correlated to word count.

Currently, only two works in literature considered the human typing rhythm associated with the production of a deceit [14, 15]. Grimes et al. tested the Keystroke Dynamics Deception Detection model with a pilot study. Each subject shared three statements, two truthful and one falsehood, on a web page. Keystroke characteristics were captured by a JavaScript based web application. The authors did not discuss the results, but they pointed out its limitations. Banerjee et al. analyzed different keystroke parameters to improve the performance of a classifier in distinguishing between truthful and deceptive writers of online reviews and essays collected via Amazon Mechanical Turk. Each Turker wrote a truthful and a deceptive text on three topics (restaurant review, gay marriage and gun control). They captured both mouse and keyboard events (KeyUp, KeyDown and MouseUp) and extracted the following features: editing patterns (i.e., number of deletion keystrokes, number of MouseUp events, number of

arrow keystrokes), writing speed and pauses (i.e., timespan of entire document, average timespan of word plus preceding keystroke, average keystroke timespan, average timespan of spaces, average timespan of non-white spaces keystrokes, average interval between words) and writing speed variations over word categories (e.g., nouns, verbs, adjectives, function words, content words). They implemented a binary SVM classifier that achieved a baseline average accuracy of 83.62%. Introducing keystroke features, authors obtained a statistically significant improvement of the deception detection classifier, ranging from 0.7% to 3.5%. We underline that none of the aforementioned work takes into account keystroke dynamics as tool to detect deceptive personal information.

4.2 Method

In this section, we present the experimental task performed by the users and the characteristics of our sample. Then, we give information about the implementation of the web site for the data collection. Finally, we describe data collected and we make our hypotheses about the users' behavior in the experimental conditions.

4.2.1 Task

We led an experiment relieving keystroke dynamics of subjects when they completed autobiographical fields. From a practical point of view, this task is similar to what happens, for example, when people complete a registration form on a social network website. This is the first reason why we decide to ask people for autobiographical information. The second one is that autobiographical information is reliable, verifiable and not depending on changes in subject's opinion or emotional state. The task follows the structure of the previous experiment described in related work section. During the experiment, participants were under to two conditions: in *true condition*, they had to complete the required fields with their real autobiographical information; while in *false condition*, they were instructed to complete the fields using false autobiographical information. These two conditions were randomized at each question. This means that subjects randomly wrote 50% of true and 50% of false autobiographical information. Subjects could distinguish the required condition by the color of field (i.e., red = false, green = true). The experiment contained 50 different autobiographical proposed to participates in both conditions (i.e., each subject compiled a total of 100 fields). In what follows, we list the 50 autobiographical fields presented to each subject:

- *Identity*: First name; Last name
- *Physical characteristic*: Gender; Weight; Height; Eye and Hair color; Shoe size.

- *Birth*: Age; Day, Month and Year of birth; Zodiac sign; Region, Province and City of birth; Citizenship; Native language.
- *Residence*: Region, Province, City, Address of residence.
- *Contacts*: e-mail; Phone number; Telephone service provider.
- *Education*: Primary school city, Middle school city, High school name, city and graduation mark, University location, department and faculty attended.
- *Interests*: Mobile phone brand and color; Car brand and color; Practiced sports; Instruments played; Animals owned; Cities where you spent a holidays and a New Year's Eve.
- *Relatives and friends*: Mother's first and last name; Father's first name; Name of a grandparent, a family member, the last partner, a close friend.

Fields were shown to participants in a random order, to avoided that the user could guess the next question. As an example, on-line registration forms usually asks user's surname right after his name, so the user could infer the information he has to insert next.

Before starting the real experiment, participants were required to complete a warm-up block (consisting of six not autobiographical fields) to make the subject takes confidence with the task. The purpose of this block was to make sure that the subject understands the task and takes confidence with the tool. The warm-up block did not contain autobiographical information, but sentences related to the actual subject's situation or his/her preferences. Data collected from this block was not taken into account for the statistical analysis. The total duration of the task was about 15-20 minutes.

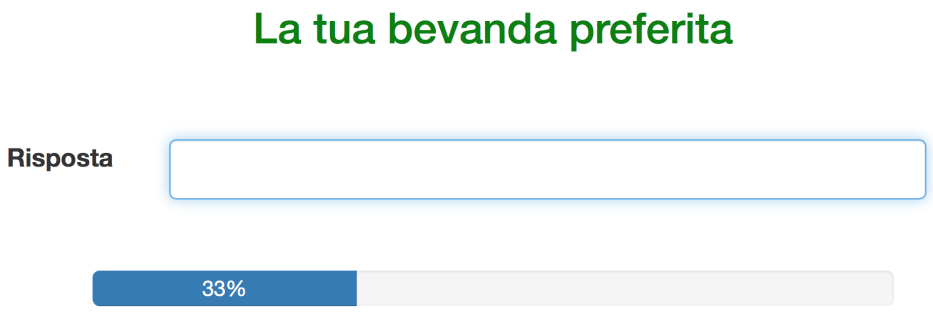


Figure 4.1: User Interface.

In order to implement the task, we designed a website using PHP, HTML, MySql and JavaScript. In particular, the recording of keystrokes and timing

was implemented using JavaScript. Figure 4.1 illustrates the final website user interface. Participant has to insert the answer in the provided rectangular field. The green color of the autobiographical field indicates that a true answer is required, while the red color of the autobiographical field indicates that a false answer is required. The collected data were stored in MySql Ver 14.14 database.

4.2.2 Participants

Participants were recruited via Internet (mailing lists and Facebook invitation). In the experiment, they completed the task autonomously by connecting to our website. All subjects used their own PC to avoid the non-confidence with a specific keyboard, operative system and graphic interface that could possibly affect the ability to typing normally. Of 244 subjects recruited, 204 subjects completed the experiment, 31 had quit it before the end and 9 had stop after registration form. Subjects with visual diseases that could affect the test performance were excluded (e.g., dyschromatopsia, aniridia). Not Italian mother tongue subjects that complete the experiment with a non-Italian keyboard were ruled out. According to a first observational data analysis, we leave out participants that showed a clear intention to compromise the test (e.g., Name answer "Goofy"). The final sample counted 190 participants, 145 females and 45 males (age mean = 39.65, SD = 12.76, years of education mean = 16.45, SD = 1.8). Before the registration at our web site, all subjects agreed with the informed consent. Finally, they compiled a brief form containing demographic information, such as age, education, job, mother tongue and visual disease, before receiving task instructions. Since each subject completed 50 true and 50 false sentences, we collected $19,000$ total observation, respectively $9,500$ true and $9,500$ false.

4.2.3 Collected Measures

Because previous studies on user authentication deemed that a timing resolution from $0.1s$ to $1\mu s$ is sufficient to capture typing characteristics, data were time stamped and measured up to microseconds (μs) precision. During each experimental session, we collected the following data:

- Prompt-first digit index: interval between the onset of the stimulus and the first key pressed.
- Prompt-enter index: total time between onset of the stimulus and the enter key pressed.
- Down time: timestamp for each key pressed.
- Up time: timestamp for each key released.

- Character frequency: assigning an identification code to any key, the frequency of pressing a specific character was recorded.

From raw data, the following measures have been calculated:

- Writing time: speed of typing, that correspond to prompt-enter index divided by the number of characters typed.
- Up and down time: sum of down and up time for each key pressed.
- Press time: time duration between a key down and a key up.
- Flight time: interleaving time between a key up and a key down.
- Tri-graphs: sum of up time, down time or up and down time for three consecutive keys.
- Special characters frequency: number of key pressing for Shift, Del and Canc, Space and Arrows characters.
- Answer length: number of final character of the answer.

4.2.4 Hypothesis

Based on literature evidence, we expected that people show a different keystroke dynamics pattern when they mislead and when they write true information. In particular, we expected that: (i) Subjects take a longer time to start writing personal information when they are declaring the false; (ii) a greater total time is required to write a false information than a true information; (iii) subjects use a larger number of correction keys (e.g., delete, canc) when they have to produce fake information; (iv) greater number of characters is used in responding the truth; (v) participants show a different keystroke pattern when they are writing the false and when they are giving true information. Specially, we assume that in lying condition there is a lengthening in press time and flight time.

4.3 Data Analysis and Models

In this section, we discuss the mathematical tool and models we used to analyze data obtained from users. After collecting data, the data analysis procedures features extraction, answers selection, features selection, data filtering, data normalization, and classification are detailed explained in the following sections.

4.3.1 Features Extraction

From the raw data collected during the task, we have extracted features in order to build a dataset that can be used for classification. Data related to keystrokes was extracted from the actual user input with her keyboard. Differently from other kind of features, the raw data of keystrokes are numeric series with different lengths according to the number of key pressed. In order to use this data into a classifier, we needed to process this data with variable to obtain a vector of features with fixed number of elements. For this reason, we extracted from each keystrokes data (up time, down time, press time and flight time, and tri-graphs) several statistical features: minimum, maximum, mean, median, standard deviation, median absolute deviation, variance, kurtosis, and skewness.

4.3.2 Answers Selection

In order to build the classification model, an answer selection was run. First, responses that contained too few characters to capture typing pattern have been removed (Age, Day of birth, Weight, Height, Shoe size). Second, the autobiographical fields for which the majority of the subjects gave an inconsistent response have been cut out (Name of your high school, High school graduation, Name of your university department). A final number of 42 autobiographical fields were considered for statistical analysis. Finally, to investigate if some answers are better than others in revealing differences in the keystroke pattern between truth and lies, a point biserial correlation analysis was run, considering 10 main features as dependent variables.

4.3.3 Features Selection

Features considered are the following: (1) prompt-first digit index, (2) writing time, (3) mean of up and down time, (4) mean of up time, (5) mean of down time, (6) mean of flight time, (7) mean of press time, (8) tri-graph mean of up and down time, (9) tri-graph mean of up time, (10) tri-graph mean of down time. Table 4.1 reports the rpb coefficient for autobiographical fields that showed a higher correlation value between the true/false condition and the dependent variables (rpb > 0.5 = large effect size, rpb > 0.3 = medium effect size, and rpb < 0.1 = small effect size). Since our feature space consists of more than three hundred features, the classification could be affected by a phenomenon known as curse of dimensionality [112]. In order to avoid this phenomenon, we performed a feature selection using an ANOVA F-value between labels and features for classification tasks. In practice, this features selection gives score to each feature according to its correlation with the label. Thus, we select one hundred features with the best correlation score.

Table 4.1: Rpb coefficient for autobiographical fields.

Features	First name	Last name	Year birth	e-mail	Phone number
(1)	**0.57**	**0.61**	**0.32**	**0.66**	0.07
(2)	**0.26**	**0.29**	**0.43**	**0.65**	**0.23**
(3)	**0.34**	**0.23**	**0.47**	**0.62**	**0.31**
(4)	**0.34**	**0.23**	**0.48**	**0.61**	**0.32**
(5)	**0.34**	**0.24**	**0.48**	**0.62**	**0.31**
(6)	**0.28**	**0.20**	**0.47**	**0.59**	**0.34**
(7)	-0.01	0.03	-0.01	0.09	-0.02
(8)	**0.34**	**0.25**	**0.47**	**0.53**	**0.29**
(9)	**0.33**	**0.24**	**0.46**	**0.61**	**0.34**
(10)	**0.33**	**0.25**	**0.46**	**0.61**	**0.33**

4.3.4 Data Filtering

As we described in Section 4.2, the data collection has been performed in an unsupervised environment, in order to simulate a real world scenario. Unfortunately, such method is prone to human errors or distractions. For this reason, we filtered out every sample with anomalous durations of features prompt-first digit and prompt-enter. In particular, we discarded the sample with the values for those features above the 90% according to the dataset (i.e., around 5 and 20 seconds for prompt-first digit and prompt-enter, respectively). Reasonably, if we can measure durations greater than those bounds, it means that the participant is distracted, thus her performance for that answer should not be considered valid.

4.3.5 Data Normalization

Since the behavior of a user while she is answering to questions changes from subject to subject, we had to consider to normalize the data collected from a user. To normalize data, we applied a transformation of each value in z-point, based on the mean and the standard deviation of all the answers of the specific user. With such normalization, it is possible to reduce the difference between users with very different behaviors and compare them. This normalization can be applied to a specific user only if we collected the behavior of that specific participant under both conditions (i.e., true and false). Moreover, such normalization can be done only on features regarding times and keystrokes. The normalization was not applied to features such as the frequency of use for special characters and answer length because they are related to the answer content instead of users' behavior. Finally, we scaled from 0 to 1 the values applying a min-max scaler for each feature in

the dataset. This scaling is useful to provide the classifier with a fixed range of values for the features, thus to speed up the training and the validation.

4.3.6 Classification

Random forest classifier is an ensemble supervised machine learning algorithm that acquires knowledge about a specific context through examples. After the training phase, where such classifier makes up their knowledge from past experience, it produces an inferred model able to classify unlabeled examples. In an optimal test scenario, the algorithm determines properly the class labels for unseen examples. The main principle behind ensemble methods is that a group of weak learners can be combined together to form a strong learner. In our case, Random forest leverages decision tree classifiers as weak learners. In practice, this ensemble method combines together the results of several decision trees trained with different portions of the training dataset and different subsets of features. Interested readers could find additional details in [113].

In our analysis, we used a model validation technique called cross-validation to evaluate the performance of our classifier. The performance can be generalized over different and independent training and test sets. This technique tries to simulate the performance of the predictive model on the real world scenarios and limit problems like over fitting. For each round, cross-validation creates a partition of the dataset into two complementary subsets given certain constraints by the operator, trains the model on one subset (i.e., training set), and tests the model on the other subset (i.e., testing set). The cross-validation provides more multiple rounds on different partitions of the dataset, and the results are averaged over all the rounds.

4.4 Results

In this section,we report and discuss the results obtained both from statistical analysis and the classification model introduced in the previous section. First, we discuss statistical results. Second, we describe the possible scenarios of application of our deceit detection analysis. Then, for each scenario, we focus on the settings of the mathematical models (i.e., normalization and cross validation) that have to be used in that scenario.

4.4.1 Statistical Analysis

We analyzed the statistical differences in collected data between the two task conditions (true condition and false condition). The analysis of the mean speed difference between true and false condition for each subject, reveals that participants are generally slower in lying than telling the truth. This difference emerges in all of the 10 features considered but press time.

Table 4.2 reports the percentage of participants that are averagely slower in true condition than false condition and the results for the pair-samples t-test analysis. In *Mean time difference* column, we report the participants that are slower in the true condition than in the false one. Mean of the time difference between true and false answers is also reported in percentage. In last column are reported the value for pair-samples t-test, all effects are reported as significant at $p < .01$. The prompt-first digit index is the best feature that discriminates between the two conditions.

Analyzing answer length, we found that 90% participants type averagely a larger number of characters answering in true condition than in the false one. This difference is statistically significant ($t = 16.76$, $p < 0.01$). As regards the number of special characters pressed, 70.5% of subjects use a greater number of space breakers in true condition comparing to the false one ($t = 9.71$, $p < 0.01$). Conversely, participants have a statistically lower rate of use of Shift ($t = -4.30$, $p < 0.01$) and Del and Canc ($t = -3.61$, $p < 0.01$) keys in true condition compared to false condition. No difference between the two conditions emerges in the frequency of use for other special characters.

Table 4.2: Statistical analysis results.

Features	**% of subjects**	**Mean time difference (%)**	**t-test**
(1)	98.42%	24.83%	t=-23.96, $p < .01$
(2)	73.16%	10.91%	t=-7.03, $p < .01$
(3)	72.63%	9.07%	t=-6.93, $p < .01$
(4)	73.68%	10.11%	t=-7.58, $p < .01$
(5)	73.16%	10.19%	t=-7.56, $p < .01$
(6)	73.68%	13.52%	t=-7.09, $p < .01$
(7)	44.21%	-0.82%	t=0.87, $p = .38$
(8)	73.16%	8.34%	t=-7.35, $p < .01$
(9)	74.74%	10.22%	t=-7.76, $p < .01$
(10)	74.74%	10.31%	t=-7.77, $p < .01$

4.4.2 Scenarios

We consider three main scenarios when it is required to understand if the user is answering the truth to a question or if she is providing false information. Each scenario is characterized by the information available to train the classifier. In what follows, we list three scenarios:

- *New answer*: the participant answers to a single question and we want to classify her answer. This means that we can use as training set both the other user answers and the answers of other users in the dataset.

In this scenario, it is possible to apply the normalization over the user's answers and perform a leave-one-out cross-validation.

- *New question*: all users answer to a question never asked before. In this case, it is possible to train the classifier with a training set including all the questions previously answered by subjects. Once again, it is possible normalize the data in the training set according to the user that is answering since we know her behavior from the previously answered questions. In this scenario, we can apply a leave-one-question-out cross-validation, where for each turn the data about a specific question are excluded from the training set to become the test set.

- *New user*: in this scenario, we aim to evaluate if a new user is lying or not during the whole task. This scenario is useful in order to understand if the user's behavior in both the test conditions is similar to the ones of the other population in the dataset. The cross-validation used to simulate this scenario consists in training a classifier with the data from all the users of the population except for one, which becomes part of the test set.

For each scenario, we can consider two cases: if we have the enrollment phase of a registration or not. This because during the enrollment phase the user answers on truthful and deceit conditions, thus we normalize the data collected from the user. In the other case, we do not have any prior information about the user behavior in both conditions. In fact, the new user might answer only under a specific condition (e.g., she answers only the truth), thus we cannot apply any normalization on data.

4.4.3 Classification Results

In this analysis, we considered a dataset of 1580 examples (i.e., answers) and we evaluated the performance of the classification leveraging cross-validation on different partition on the dataset according to the scenario mentioned above. More in details, we performed a leave-one-out cross-validation for new answer scenario, and a leave-one-label-out cross-validation for new question and new user, where the label corresponds to question and user identifiers, respectively. For each scenario, we also evaluated the influence of normalization application on user data. The results of our evaluation are summarized in Figure 4.2, where the accuracy is expressed in terms of F-measure for the former scenario (i.e., new answer) and averaged F-measure for the latter two scenarios (i.e., new question and new user).

As we can notice, the results for the scenarios where normalization of user data is applied outperform the ones where it is not applied. We recall that normalization can be applied only if we are considering data collected from a user under both conditions, thus we know the behavior of the user

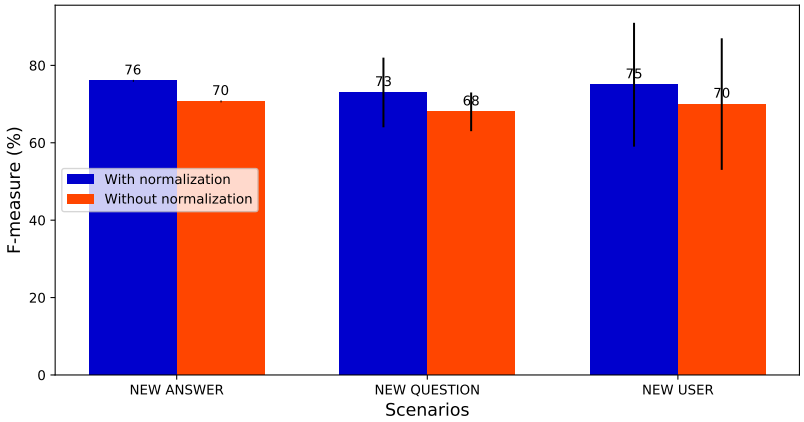


Figure 4.2: Accuracy for the three scenarios.

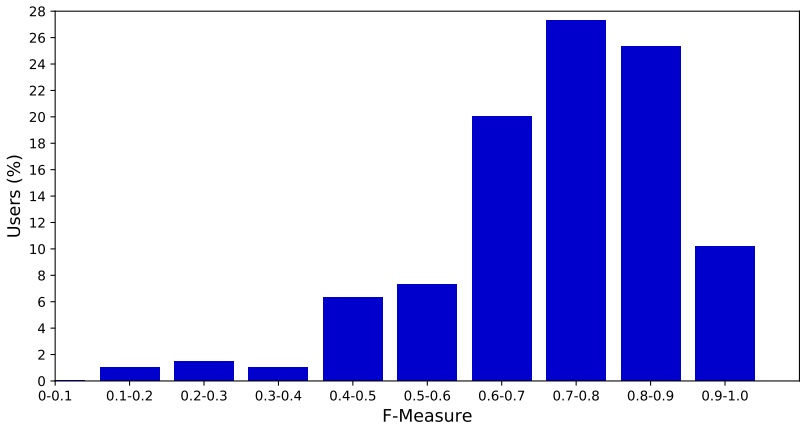


Figure 4.3: Frequency distribution of classification in *new user* scenario, with normalization.

when she lies or if she tells the truth. In particular, we are able to achieve an accuracy of 76% on the new answer scenario with normalization. We can also observe a significant standard deviation on new user scenario. In order to investigate further on this phenomenon, we report in Figure 4.3 the frequency of F-measure obtained by cross-validation runs for single users. We observe that for the majority of users we obtain an accuracy greater than 60%. This means that our model is not able to classify correctly (i.e., the accuracy is under the 50%) only the 4% of the whole population.

We can also observe a significant standard deviation on new user scenario, both with and without normalization. In order to investigate further

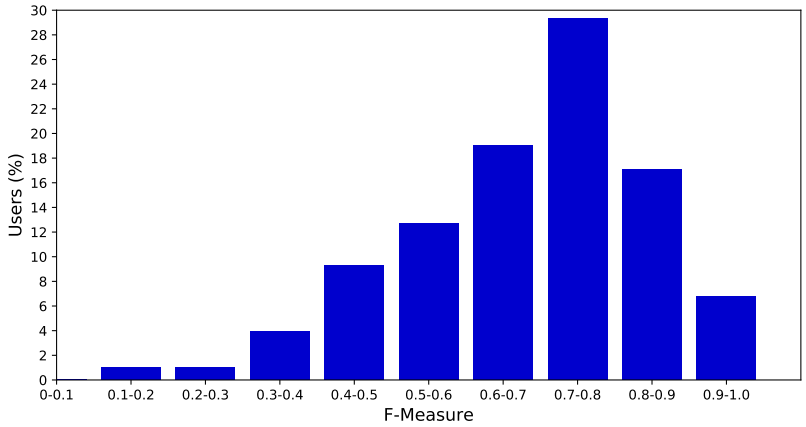


Figure 4.4: Frequency distribution of classification in *new user* scenario, without normalization.

on this phenomenon, we report in figures 4.3 and 4.4 the frequency of F-measure obtained by cross-validation runs for single users with and without normalization, respectively. We observe that, in both graphs, for the majority of users we obtain an accuracy greater than 60%. This means that our model is not able to classify correctly (i.e., the accuracy is under the 50%) only the 4% and 7.6% of the whole population, with and without applying the normalization, respectively.

4.5 Discussion

From our analysis, we were able to make several interesting observations. Subjects have slower writing pattern when they are in a position to provide fake information. In particular, they manifest a lengthening in the total writing speed that is due to more time take in down time, up time and flight time. Furthermore, in the falsehood condition participants take a longer time to start writing (prompt-first digit index). The origin of the strong importance of this feature can be certainly explained by the cognitive mechanisms involved in the processing of the lie, discussed in Section 4.1.

Another important observation concerns the answers that better show the difference between a true and a false condition response. The autobiographical information that has a higher correlation with the experimental condition are frequently typed on the keyboard. Therefore, the shortest response time in answering truthfully could be a result of a procedural memory mechanism [114]. According to these results, we can hypothesize that the characteristic keystroke pattern during the production of a deceit response is due to two different processes: a cognitive mechanism that results in a

longer time before start writing, and a procedural mechanism that makes more automatic the typing of information that we use frequently.

4.6 Summary

In this chapter, we presented a new method based on keystroke dynamics to distinguish between fake and truthful personal information written via a computer keyboard. We reached an overall accuracy of 76% in the classification of a single answer as truthful or deceptive. Compared with the current existing behavioral lie detection tools, as the CIT and the aIAT [95, 96], and similar to mouse dynamics lie detection [98], keystroke dynamics offers the advantage to be apply as covert lie detection, as required by real context (e.g. in social account creation or in online banking authentication). Moreover, the detection of false information via keystroke dynamics is more flexible than mouse dynamics. Summarizing, our work shows that keystroke analysis could have a great potential in detecting the veracity of self-declared personal information. We believe our work paves the way for further research in this area, as well as for a significant number of useful practical applications in all scenarios where users input information remotely via a keyboard.

Part II

HCI Impact on Network Traffic

Chapter 5

Analysis of Network Traffic on Mobile Devices

The last decade has been marked by the rise of mobile devices which are nowadays widely spread among people. The most diffused examples of such mobile devices are the smartphone and the tablet. When compared with traditional cell phones, smartphones and tablets (we referred as *mobile devices*) have an enormously increased computational power, more available memory, a larger display, and the Internet connectivity via both Wi-Fi and cellular networks. Moreover, such devices run mobile operating systems which are able to experience multimedia contents, as well as to run mobile applications (also called *apps*). Combined together, these elements enable both smartphones and tablets to have the same functionalities typically offered by laptops and desktop computers.

According to the report in [115], smartphone users were 25.3% of the global population in 2015, and this percentage is expected to grow till 37% in 2020. Similarly, the statistics about tablets reported in [116] indicate a global penetration of 13.8% in 2015, expected to reach 19.2% in 2020. The driving forces of this tremendous success are the ubiquitous Internet connectivity, thanks to the worldwide deployment of cellular and Wi-Fi networks, and a large number of apps available in the official (and unofficial) marketplaces. A mobile device typically hosts a lot of sensitive information about its owner, such as contacts, photos and videos, and GPS position. Such information has to be properly protected, especially when it is transmitted to remote services. Since an important fraction of the overall Internet traffic is due to mobile devices, it is not surprising that attackers and network traffic analysts have soon started to target them.

Network traffic analysis (we referred as *traffic analysis*) is the branch of computer science that studies inferential methods which take the net-

work traces of a group of devices as input, and give information about those devices, their users, their apps, or the traffic itself as output. Network traces can be captured at different layers (e.g., data-link layer, application layer), different points (e.g., within a Wi-Fi network, within the devices), and their content is often encrypted (making analysis even more challenging). Typically, researchers follow two different approaches for analyzing mobile network traffic: (i) taking pre-existent methods designed for traditional Internet traffic, and adapting them to the mobile scenario; or (ii) developing new methods tailored to mobile Internet traffic properties. For this reason, the research community investigates on network traffic analysis techniques to improve both security and privacy on mobile devices.

Chapter Organization. The rest of the chapter is organized as follows. In Section 5.1, we present the classifications adopted to report the works. In the following sections, we survey the works according to three criteria: (i) the goal of the analysis; (ii) the capturing point; and (iii) the targeted mobile platforms. Especially, in Section 5.2, we explain the goal of analysis targeted on the human interaction with mobile devices. Finally, we conclude this chapter in Section 5.3.

5.1 Categorization of Work

Overall, we survey 56 works, published between 2010 and 2017. Figure 5.1 shows that the number of publications in the considered research field has significantly increased in the last years. We believe that this amount of work

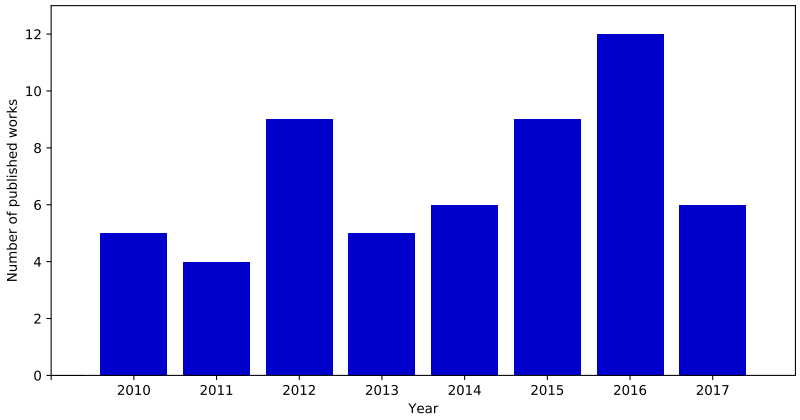


Figure 5.1: Number of published works sorted by year.

will grow in the future as the global spreading of mobile devices is increasing

and their contribution to the worldwide Internet traffic is becoming more significant.

In Table 5.1, 5.2, we report the surveyed works according to these criteria. Moreover, for each work we indicate whether the proposed analyses are still applicable in case of traffic encryption via either SSL/TLS or IPsec. It is worth to notice that a few works (i.e., Wei et al. [117], and Tadrous and Sabharwal [118]) propose multiple traffic analysis techniques, each affected by traffic encryption in a different way.

Table 5.1: All surveyed works.

Year	Paper	**Goal of Analysis**	**Point of Capturing**	**Targeted Mobile Platform**	**SSL/TLS**	**IPsec**
2010	Afanasyev et al. [119]	Characterization, Usage	APs, Wired	Platform-independent	✗	✗
	Falaki et al. [120]	Characterization, Usage	Devices	Android, Windows Mobile	✔	✗
	Husted et al. [121]	Positioning	Simulator	Platform-independent	✔	✔
	Maier et al. [122]	Characterization, Usage	Wired	Platform-independent	✗	✗
	Shepard et al. [123]	Characterization	Devices	iOS	✔	✗
2011	Finamore et al. [124]	Characterization, Usage	Wired	Platform-independent	✗	✗
	Gember et al. [125]	Characterization, Usage	APs	Platform-independent	✗	✗
	Lee et al. [126]	App	Wired	Android, iOS	✗	✗
		Characterization, Usage		Platform-independent	✗	✗
	Rao et al. [127]	Characterization	Wired	Android, iOS	✗	✗
2012	Baghel et al. [128]	Characterization	Wired	Android	✔	✗
	Chen et al. [129]	Characterization	Wired	Platform-independent	✗	✗
	Ham et al. [130]	Usage	Devices	Android	✔	✔
	Musa et al. [131]	Trajectory	Monitors	Platform-independent	✔	✔
	Shabtai et al. [132]	Malware	Devices	Android	✔	✔
	Stevens et al. [133]	PII Leakage	APs	Android	✗	✗
	Su et al. [134]	Malware	Devices	Android	✔	✗
	Wei et al. [135]	Malware	Wired	Android	✗	✗
	Wei et al. [117]	Characterization	Devices	Android	✔	✔/✗
2013	Barbera et al. [136]	Sociological	Monitors	Platform-independent	✔	✔
	Kuzuno et al. [137]	PII Leakage	Devices	Android	✗	✗
	Qazi et al. [138]	App	APs, Devices	Android	✔	✗
	Rao et al. [139]	App, PII Leakage	Wired	Android, iOS	✔	✗
	Watkins et al. [140]	User Actions	APs	Android	✔	✔
2014	Chen et al. [141]	OS	APs	Android, iOS	✔	✗
		Tethering	Monitors, Wired	Platform-independent	✔	✗
	Coull et al. [142]	User Actions, OS	Devices	iOS	✔	✗
	Crussell et al. [143]	Ad Fraud	Emulators	Android	✗	✗
	Lindorfer et al. [144]	Characterization	Emulators	Android	✗	✗
	Shabtai et al. [145]	Malware	Devices	Android	✔	✔
	Verde et al. [146]	User Fingerprinting	Wired	Platform-independent	✔	✔

Table 5.2: All surveyed works.

Year	Paper	**Goal of Analysis**	**Point of Capturing**	**Targeted Mobile Platform**	**SSL/TLS**	**IPsec**
2015	Chen et al. [147]	Characterization	Wired	Android	✗	✗
	Fukuda et al. [148]	Characterization, Usage	Devices	Android, iOS	✔	✔
	Le et al. [149]	App, PII Leakage	Devices	Android	✔	✗
	Park et al. [150]	User Actions	Wired	Android	✔	✗
	Soikkeli et al. [151]	Usage	Devices	Platform-independent	✔	✔
	Song et al. [152]	PII Leakage	Devices	Android	✔	✗
	Wang et al. [153]	App	Monitors	iOS	✔	✔
	Yao et al. [154]	App	APs, Emulators	Android, iOS, Symbian	✗	✗
	Zaman et al. [155]	Malware	Devices	Android	✗	✗
2016	Alan et al. [156]	App	APs	Android	✔	✔
	Conti et al. [45]	User Actions	Wired	Android	✔	✗
	Mongkolluksamee et al. [157]	App	Devices	Android	✔	✗
	Narudin et al. [158]	Malware	Devices, Emulators	Android	✗	✗
	Nayam et al. [159]	Characterization	Wired	Android, iOS	✗	✗
	Ren et al. [160]	PII Leakage	Wired	Android, iOS, Windows Phone	✔	✗
	Ruffing et al. [161]	OS	Monitors	Android, iOS, Windows Phone, Symbian	✔	✔
	Saltaformaggio et al. [162]	User Actions	APs	Android, iOS	✔	✔
	Spreitzer et al. [163]	Website Fingerprinting	Devices	Android	✔	✔
	Tadrous et al. [118]	Characterization	APs	Android, iOS	✔	✔/✗
	Vanrykel et al. [164]	PII Leakage, User Fingerprinting	Wired	Android	✗	✗
	Wang et al. [165]	Malware	Wired	Android	✔	✗
2017	Arora et al. [166]	Malware	Devices	Android	✔	✔
	Continella et al. [167]	PII Leakage	Wired	Android	✔	✗
	Espada et al. [168]	Characterization	Devices	Android	✔	✗
	Malik et al. [169]	OS	APs	Android, iOS, Windows Phone	✔	✔
	Taylor et al. [170]	App	Wired	Android	✔	✗
	Wei et al. [171]	Characterization, Usage	Wired	Platform-independent	✗	✗

5.1.1 Classification by Goal of the Analysis

The first classification takes into account the goal of the analysis performed on the captured mobile traffic. For each surveyed work, Table 5.1 and 5.2 provides this information in the *Goal of the Analysis* column.

Overall, we are able to identify fourteen goals. In Figure 5.2, we depict such goals by their field of pertinence: apps, mobile users, and mobile devices. In what follows, we list and briefly describe each of the goals:

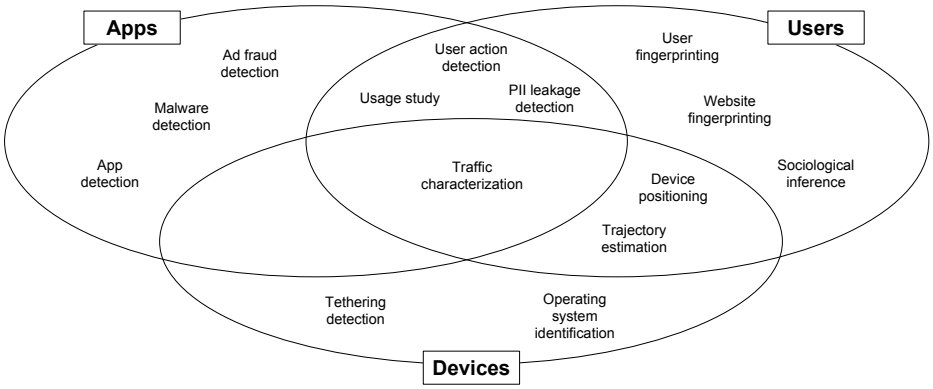


Figure 5.2: The goals of traffic analysis targeting mobile devices, and their pertinence to apps, mobile users, and mobile devices.

1. **Ad fraud detection** (*Ad Fraud* in Table 5.1 and 5.2): to detect ad fraud by a mobile app, i.e., to recognize if a mobile app is trying to trick the advertising business model (e.g., fabricating false user clicks on ads). This type of analysis is fundamental for ad providers, which can protect themselves from dishonest app developers trying to illicitly earn money.

2. **App identification** (*App* in Table 5.1 and 5.2): to recognize the network traffic belonging to a specific mobile app. This type of analysis can help network administrators in resource planning and management, as well as in app-specific policy enforcement (e.g., forbidding a social network app within an enterprise network). Moreover, app identification can be employed to uncover the presence of sensitive apps (e.g., dating, health, religion) in the mobile device of a target user.

3. **Device positioning** (*Positioning* in Table 5.1 and 5.2): to discover the geographical position of a mobile device. This type of analysis helps infer social status, interests, and habits of the owner of a mobile device. As a further step, the profiles of several mobile users can be aggregated for marketing, as well as sociological studies. See Section 5.2.1.

4. **Malware detection** (*Malware* in Table 5.1 and 5.2): to detect whether a mobile app behaves maliciously (e.g., downloading and installing malicious code from the network). This type of analysis can be used to assess the security of an app submitted by a developer to a mobile marketplace. In such case, the result of the security tests decides whether the app can be released to the public. Moreover, malware detection algorithms can be embedded into anti-virus apps that mobile users can use to check whether an installed app is malicious.

5. **Operating system identification** (*OS* in Table 5.1 and 5.2): to discover the operating system of a mobile device. This type of analysis is usually a preliminary phase for more advanced attacks against mobile devices: the adversary tries to infer the operating system of the target mobile device in order to subsequently exploit an ad-hoc vulnerability for that specific OS. Moreover, operating system identification carried out on a large mobile user population can be a starting point for other types of analysis not directly related to computer science (e.g., sociological studies).

6. **PII leakage detection** (*PII Leakage* in Table 5.1 and 5.2): to detect and/or prevent the leakage of a mobile user's Personal Identifiable Information (PII). This type of analysis can be employed to assess the behavior of a mobile app from a privacy point of view, by checking which PII it actually discloses to remote hosts. Detecting PII leakage is also the first step to prevent such problem, since it is then possible to block network transmissions carrying PII, or replace sensitive information with bogus data. See Section 5.2.2.

7. **Sociological inference** (*Sociological* in Table 5.1 and 5.2): to infer some kind of sociological information about mobile users (e.g., language, religion, health condition, sexual preference, wealth), from one or more properties related to their mobile devices (e.g., list of installed apps, associated Wi-Fi networks). See Section 5.2.3.

8. **Tethering detection** (*Tethering* in Table 5.1 and 5.2): to detect if a mobile device is tethering, i.e., it is sharing its Internet connectivity with other devices, for which it acts as an access point. Tethering constitutes a problem for cellular network providers, since it significantly increases the volume of network traffic generated by a single client. Such providers are therefore interested in tethering detection techniques that can be used to prevent their customers from sharing their Internet connectivity, or simply require them to pay an extra fee to do that.

9. **Traffic characterization** (*Characterization* in Table 5.1 and 5.2): to infer the network properties of mobile traffic. The knowledge of such

properties is crucial to effectively deploy and configure the resources in cellular networks, as well as in Wi-Fi networks serving mobile devices. See Section 5.2.4.

10. **Trajectory estimation** (*Trajectory* in Table 5.1 and 5.2): to estimate the trajectory (i.e., the movements) of a mobile device within a geographical area. This type of analysis can help study the interests and social habits of a mobile user. Moreover, trajectory estimation can aid road traffic prediction along urban streets, by leveraging the most frequent trajectories followed by the citizens that move along the city. See Section 5.2.5.

11. **Usage study** (*Usage* in Table 5.1 and 5.2): to infer the usage habits of mobile users (e.g., which are the most frequently used apps). As an example, the knowledge of the places where mobile devices are mostly used can drive the deployment of cellular stations and Wi-Fi hotspots. See Section 5.2.6.

12. **User action identification** (*User Actions* in Table 5.1 and 5.2): to identify a specific action that a mobile user performed on her mobile device (e.g., uploading a photo on Instagram), or to infer some information about that specific action (e.g., the length of a mobile user's message sent through an instant messaging app). Researchers can employ such analysis to discover the identity behind an anonymous social network profile. This can be accomplished by verifying if there is a match between the events reported on that profile's page, and the actions a suspect performed while using the mobile app of that social network. Alternatively, it is possible to build behavioral profiles of mobile users, which are useful for user reconnaissance within networks and, in aggregated form, for marketing studies. See Section 5.2.7.

13. **User fingerprinting** (*User Fingerprinting* in Table 5.1 and 5.2): to detect the traffic belonging to a specific mobile user. This type of analysis can be employed to trace a mobile user, by approximating her position with the location of the Wi-Fi hotspot or cellular station to which her mobile device is connected. From this information, it is then possible to build a behavioral profile of that mobile user. Alternatively, it is possible to examine a mobile traffic dataset in order to extract and group together the network traces generated by a specific mobile user. Such data can be subsequently used for other types of traffic analysis targeting that user. See Section 5.2.8.

14. **Website fingerprinting** (*Website Fingerprinting* in Table 5.1 and 5.2): to infer which websites and/or webpages are visited by a mobile user while navigating via the web browser of her mobile device.

Similarly to sociological inference, this type of analysis can reveal interests, social habits, religious belief, as well as sexual and political orientations of a mobile user. See Section 5.2.9.

5.1.2 Classification by Point of Capturing

The second classification considers where and how the mobile traffic is captured. For each surveyed work, Table 5.1 and 5.2 provides this information in the *Point of Capturing* column. It is worth to notice that: (i) we are focusing on the (hardware and/or software) equipment that captures the traffic; and (ii) we report the point of capturing only for those datasets for which the authors give enough details about the collection process. Overall, we identify six different points of capturing:

- Within one or more mobile devices, i.e., client-side (*Devices* in Table 5.1 and 5.2). This type of point of capturing is particularly useful if we want to target a specific mobile app (e.g., Facebook), or a particular network interface (e.g., cellular). We specify that this category also includes the case of a network traffic logger installed within either: (i) a mobile device emulator; and (ii) a machine to which the mobile traffic is mirrored using a remote virtual network interface.

- At one or more wired network equipments (*Wired* in Table 5.1 and 5.2). The size of the population of monitored mobile devices varies according to the type of considered network equipments: thousands of mobile users in the case of edge routers (i.e., routers connecting customers to the ISP's backbone) and Internet gateways; from tens to a few hundreds in the case of VPN servers and forwarding servers (i.e., traditional desktop computer put in the middle of a wired link and set up to log all traffic traversing it).

- At one or more access points of a Wi-Fi network (*APs* in Table 5.1 and 5.2). This type of point of capturing allows the number of monitored mobile devices to vary from tens to a few thousands, and it is suitable to capture the traffic of mobile devices while their users are performing network-intensive tasks (e.g., watching streaming videos, updating apps).

- At one or more Wi-Fi monitors (*Monitors* in Table 5.1 and 5.2). Researchers usually employ this type of capturing devices to focus the network traffic collection process on a specific geographical area of interest (e.g., a train station). Such approach is often the only viable solution whether it is not possible to directly access a target mobile device, or the network to which it is connected.

- At one or more machines running virtual mobile devices, i.e., emulators (*Emulators* in Table 5.1 and 5.2). The possibility to run multiple virtual mobile devices in parallel and control them via automated tools enables a large-scale network traffic collection that would be more expensive if conducted on real mobile devices. It is important to highlight that the traffic logging is performed by the host machines or their virtualization managers. We do not consider the case in which the traffic logging takes place within the emulated mobile devices (such case is covered by the *Devices* category).

- At one or more virtual capturing points within a simulated environment generated and managed by a software program (*Simulator* in Table 5.1 and 5.2). This point of capturing can help study particular deployments of mobile devices that are not observable in a real-world scenario because of technical, economical, or legal constraints.

5.1.3 Classification by Targeted Mobile Platform

The third classification considers the mobile platforms that are targeted by the traffic analysis. For each surveyed work, Table 5.1 and 5.2 provides this information in the *Targeted Mobile Platform* column. It is worth to specify that we classify a work as *platform-independent* if its authors do not provide information about the targeted mobile platforms, or such information is not relevant to the analysis they perform on the mobile traffic.

Overall, we find four distinct mobile platforms: Android, Google's open-source mobile operating system; iOS, the operating system of Apple's mobile devices; Symbian, the first released modern mobile operating system; and Windows Mobile/Phone, the mobile counterpart of Microsoft's desktop operating system.

5.2 Goals of Traffic Analysis Targeting Mobile Devices

In this section, we survey the works according to the goal of the analysis that is performed on the mobile traffic. Table 5.3 and 5.4 summarizes the goals of the surveyed works. As shown in Figure 5.3, the most frequently pursued goal is traffic characterization (eighteen works), followed by usage study (ten works), app identification (nine works), malware detection and PII leakage detection (eight works each), user action identification (five works), operating system identification (four works), and user fingerprinting (two works). Each of the following goals counts one work only: ad fraud detection, device positioning, sociological inference, tethering detection, trajectory estimation, and website fingerprinting. As shown in Table 5.3 and 5.4, twelve works pursue two goals, and one work even three. In the following sections,

we present the goal(s) related to Human-Computer Interaction and achieved results for the surveyed work. We also discuss whether the proposed analysis works on encrypted network traffic (e.g., IPsec, SSL/TLS).

Table 5.3: The surveyed works listed by goal of the analysis.

Year	Paper	Device Positioning	PII Leakage Detection	Socio-logical Inference	Traffic Characteri-zation	Trajectory Estimation	Usage Study	User Action Identi-fication	User Finger-printing	Website Finger-printing
2010	Afanasyev et al. [119]				✔		✔			
	Falaki et al. [120]				✔		✔			
	Husted et al. [121]	✔								
	Maier et al. [122]				✔		✔			
	Shepard et al. [123]				✔					
2011	Finamore et al. [124]				✔		✔			
	Gember et al. [125]				✔		✔			
	Lee et al. [126]				✔		✔			
	Rao et al. [127]				✔					
2012	Baghel et al. [128]				✔					
	Chen et al. [129]				✔					
	Ham et al. [130]						✔			
	Musa et al. [131]					✔				
	Shabtai et al. [132]									
	Stevens et al. [133]		✔							
	Su et al. [134]									
	Wei et al. [135]									
	Wei et al. [117]				✔					
2013	Barbera et al. [136]			✔						
	Kuzuno et al. [137]		✔							
	Qazi et al. [138]									
	Rao et al. [139]		✔							
	Watkins et al. [140]							✔		
2014	Chen et al. [141]									
	Coull et al. [142]							✔		
	Crussell et al. [143]									
	Lindorfer et al. [144]				✔					
	Shabtai et al. [145]									
	Verde et al. [146]								✔	

Table 5.4: The surveyed works listed by goal of the analysis.

Year	Paper	Device Positioning	PII Leakage Detection	Socio-logical Inference	Traffic Characteri-zation	Trajectory Estimation	Usage Study	User Action Identi-fication	User Finger-printing	Website Finger-printing
2015	Chen et al. [147]				✔					
	Fukuda et al. [148]				✔		✔			
	Le et al. [149]		✔							
	Park et al. [150]							✔		
	Soikkeli et al. [151]						✔			
	Song et al. [152]		✔							
	Wang et al. [153]									
	Yao et al. [154]									
	Zaman et al. [155]									
2016	Alan et al. [156]									
	Conti et al. [45]							✔		
	Mongkolluksamee et al. [157]									
	Narudin et al. [158]									
	Nayam et al. [159]				✔					
	Ren et al. [160]		✔							
	Ruffing et al. [161]									
	Saltaformaggio et al. [162]							✔		
	Spreitzer et al. [163]									✔
	Tadrous et al. [118]				✔					
	Vanrykel et al. [164]		✔						✔	
	Wang et al. [165]									
2017	Arora et al. [166]									
	Continella et al. [167]		✔							
	Espada et al. [168]				✔					
	Malik et al. [169]									
	Taylor et al. [170]									
	Wei et al. [171]				✔		✔			

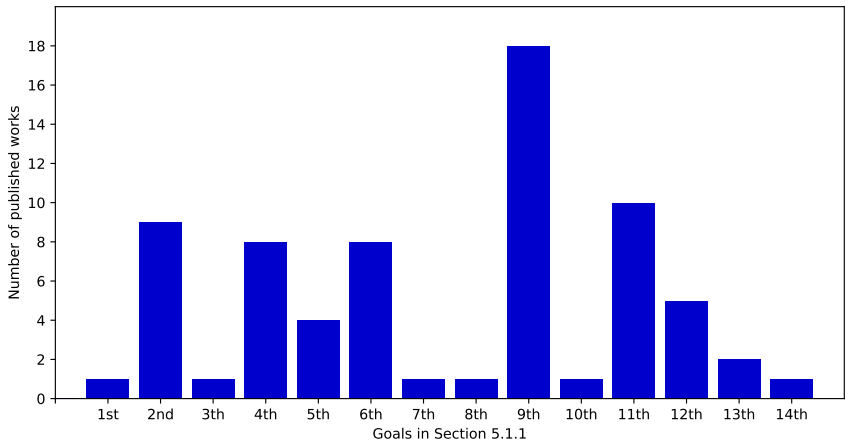


Figure 5.3: Number of published works sorted by goal of the analysis.

5.2.1 Device Positioning

The set of places frequently visited by a person tells a lot about her social status, interests, and habits. Such information can be exploited for commercial purposes (e.g., targeted advertisement), as well as intelligence activities (e.g., police investigations). Since most of the people own a mobile device and keep it with them all day long, locating the smartphone/tablet of a target user becomes a simple yet effective way to know her position. Moreover, it is possible to build a profile of the subject by aggregating multiple position detections.

We define as *device positioning* the inference of the geographical position of a mobile device by analyzing the network traffic it generates. In this section, we survey the works that propose this type of traffic analysis. We point out that we do not consider the works in which: (i) the mobile traffic is analyzed to detect the leakage of GPS coordinates (we consider this kind of works in Section 5.2.2); (ii) the movements (not the position) of a mobile device are reconstructed from its network traffic (we review this kind of works in Section 5.2.5); and (iii) the analysis performed on the network traffic is device-agnostic, i.e., it does not take into account the fact that the target devices are smartphones/tablets (this kind of works is excluded since it is too generic).

Husted and Myers in [121] investigate whether a malnet (i.e., a colluding network of malicious Wi-Fi devices) can successfully determine the location of a mobile device. Each malicious node looks for probe requests carrying the MAC address of the target mobile device, and uploads its findings to a central server where the data coming from all nodes is used for tri-

lateration. Through a software simulation of a metropolitan population of users equipped with 802.11g mobile devices, the authors show that 10% of tracking population is sufficient to track the position of the remaining users. Besides, the tracking benefits from extending the broadcasting range of the mobile devices. This suggests that the adoption of newer 802.11 standards can make it feasible to build a geolocating malnet.

5.2.2 PII Leakage Detection

A mobile device is a source of sensitive information about its owner (e.g., phone number, contacts, photos, videos, GPS position). In addition, apps often require to access such information to deliver their services. As an example, an instant messaging app (e.g., WhatsApp, Telegram, WeChat) requires to access the contacts saved in the device's address book. As another example, a social network app (e.g., Facebook, Instagram) requires to inspect the device's memory to find photos.

To disclose sensitive information to a remote host, an app must be authorized to: (i) access some kind of sensitive information (e.g., the GPS position); and (ii) connect to the Internet. The disclosure can be either allowed or illicit, depending on the level of sensitivity of the disclosed information, the reason why the app transmits such information to a remote host, and whether the user is aware of this transmission of sensitive data.

In this section, we focus on *personal identifiable information* (*PII*), which is information that can be used to identify, locate, or contact an individual. In the domain of mobile devices, there are four types of PII:

- Information related to mobile devices, such as IMEI (International Mobile Equipment Identity, a unique identifier associated to each mobile device), Android Device ID (an identifier randomly generated on the first boot of Android device), and MAC address (a unique identifier assigned to each network interface).

- Information related to SIM cards, such as IMSI (International Mobile Subscriber Identity, a unique identifier assigned to each subscriber of a cellular service), and SIM Serial ID (the identifier assigned to each SIM card).

- Information related to users, such as name, gender, date of birth, address, phone number, and email.

- Information about user's location, such as GPS position and ZIP code.

We define as *PII leakage detection* the analysis of the network traffic of a mobile device in order to detect the leakage of user's PII. Once a PII leakage is detected, it is possible to apply suitable countermeasures, such as blocking the network flows carrying the PII, or substituting the sensitive

information with bogus data. The latter approach is a good solution for mobile users who want to protect their privacy while being able to enjoy the functionalities of the apps.

In Table 5.5, we present the surveyed works that deal with PII leakage detection [133, 137, 139, 149, 152, 160, 164, 167]. For each work, we summarize the targeted mobile platforms, and whether the PII leaks are simply detected or also prevented.

Table 5.5: Works that deal with PII leakage detection.

Year	Paper	Targeted Mobile Platform			Action on PII Leaks	
		Android	iOS	Windows Phone	Detection	Prevention
2012	Stevens et al. [133]	✔			✔	
2013	Kuzuno et al. [137]	✔			✔	
	Rao et al. [139]	✔	✔		✔	✔
2015	Le et al. [149]	✔			✔	
	Song et al. [152]	✔			✔	✔
2016	Ren et al. [160]	✔	✔	✔	✔	✔
	Vanrykel et al. [164]	✔			✔	
2017	Continella et al. [167]	✔			✔	

Stevens at al. in [133] present a comprehensive study on thirteen popular ad providers for Android. In particular, part of this study focuses on the analysis of ad traffic in order to detect the transmission of the user's PII. The authors observe that at the time of writing only one of the considered ad providers leverage encryption to protect its network traffic. For this reason, they choose to perform a deep packet inspection to identify the leakage of the user's private information. The results show that several types of PII (e.g., age, gender, GPS position) are leaked in clear by ad libraries. Moreover, the authors highlight that although none of the considered ad providers is able to build a complete profile of the user, the presence of UDIDs in ad-related traffic can be exploited by an external adversary to correlate sensitive information from different ad providers and build a complete user profile.

Kuzuno and Tonami in [137] investigate the leakage of sensitive information by the advertisement libraries embedded into free Android apps. They focus on both original and hashed identifiers unique to mobile devices (i.e., IMEI and Android ID) and SIM cards (i.e., IMSI and SIM Serial ID), as well as on the name of the cellular operator (CARRIER). The authors develop two components: (i) a server application; and (ii) a mobile app that can be installed on an Android device. The server application takes in input the network traffic of a set of apps that leak sensitive information, and applies a clustering method to generate traffic signatures. The mobile app leverages such signatures to identify the sensitive information leaked by the other apps installed on the device. To evaluate their solution, the authors employ the network traffic of 1,188 free Android apps and achieve the following results: 94% of HTTP messages containing sensitive information are correctly detected, with 5% false negatives (i.e., undetected HTTP messages carrying

sensitive information), and less than 3% false positives (i.e., HTTP messages without sensitive information, incorrectly identified as sensitive). Since the signature generation phase requires to inspect HTTP messages looking for sensitive information, the system cannot work on encrypted traffic.

Rao et al. in [139] and Ren et al. in [160] present ReCon, a cross-platform system that allows mobile users to control the PII leaked in the network traffic of their devices. ReCon is based on Meddle, therefore it can inspect mobile traffic even if it is encrypted at transport layer, but cannot cope with data transmissions protected by IPsec. Moreover, ReCon offers a web interface through which the user can visualize in real time which PII is leaked, and optionally modify such PII or block the connection carrying it. In [139], the authors target two mobile OSes, namely Android and iOS, and the PII leakage detection mechanism is based on a domain blacklist. In [160], also Windows Phone is considered, and PII leaks are detected using properly trained machine learning classifiers, among which the best-performing (98.1% accuracy) is a C4.5 Decision Tree. The works in [139, 160] expose an extensive leakage of sensitive information belonging to all the types of PII we listed above, as well as the transmission of usernames and passwords in both plain-text (HTTP) and encrypted (HTTPS) traffic.

Le et al. in [149] present AntMonitor, a system for collecting and analyzing network traffic from Android devices. Among other types of analysis, AntMonitor can perform PII leakage detection. The authors capture the network traffic of nine Android users for a period of five weeks, then inspect the collected dataset searching the following PII: IMEI, Android Device ID, phone number, email address, and device location. Overall, 44% and 66% of the analyzed apps leak IMEI and Android Device ID, respectively, while PII related to the user is rarely disclosed to remote hosts. It is worth to notice that the proposed analysis requires to inspect application-layer data, which is infeasible in case of traffic encryption, either at network (IPsec) or transport layer (SSL/TLS).

Song and Hengartner in [152] develop PrivacyGuard, an open-source Android app that leverages the `VPNService` class of the Android API for eavesdropping the network traffic of the apps installed on the device. The authors employ PrivacyGuard to investigate the leakage of PII related to mobile users (e.g., phone number) and devices (e.g., IMEI) by Android apps. In an evaluation conducted using 53 Android apps, PrivacyGuard detect more PII leaks than TaintDroid [80]. The proposed app can optionally replace the leaked information with bogus data. Moreover, it can inspect transmission protected by SSL/TLS (through a man-in-the-middle approach), but cannot deal with traffic encrypted via IPsec.

Vanrykel et al. in [164] investigate the leakage of sensitive identifiers in the unencrypted network traffic of Android apps. The authors develop a framework that automatically executes apps, collects their network traffic, inspects the HTTP data, and detects the identifiers that are transmitted in

clear. The analysis of 1260 Android apps (from 42 app categories) shows that: (i) the Android ID and Google Advertising ID are the most frequently leaked identifiers, while the SIM serial number, the IMSI, the device serial number, and the email of the registered Google account are less common in apps' network traffic; (ii) there is an extensive leakage of app-specific identifiers; and (iii) certain apps leak the user's phone number, email address, or position.

Continella et al. in [167] develop Agrigento, an open-source framework for the analysis of Android apps in order to detect PII leakage. Agrigento is based on differential analysis, and its workflow consists of two phases. In the first phase, the app under scrutiny is executed several times on a physical device to collect: (i) its network traffic; and (ii) additional system- and app-level information that is contextual to the execution (e.g., randomly-generated identifiers, timestamps). Subsequently, the collected information is aggregated to model the network behavior of the app. In the second phase, a specific PII within the operating system of the mobile device is set to a different value. The app is then executed once again to collect its network traffic and the contextual information. Finally, a PII leakage is reported if the collected data does not conform to the model learned before. Evaluated on 1,004 Android apps, Agrigento detects more privacy leaks than currently available state-of-the-art solutions (e.g., ReCon [160]), while limiting the number of false positives. The proposed framework requires to inspect HTTP messages and leverages a man-in-the-middle approach to deal with HTTPS traffic. However, Agrigento does not work on network traffic encrypted via IPsec.

5.2.3 Sociological Inference

A property of a mobile device (e.g., the list of installed apps, the Wi-Fi networks to which the device associated) characterizes its owner. Sociologists can leverage this kind of information to study a population of mobile users. In this section, we review the works that deal with *sociological inference*, which we define as the analysis of the network traffic generated by mobile devices in order to infer some kind of sociological information about their users.

Barbera et al. in [136] investigate whether sociological information about a large crowd can be inferred by inspecting the Wi-Fi probe requests generated by the mobile devices of those people. First of all, the authors: (i) devise a methodology to convert a dataset of Wi-Fi probe requests into a social graph representing the owners of the monitored mobile devices; and (ii) develop an automatic procedure to infer the language of a given SSID. Subsequently, they target gatherings of people at urban, national, and international scale, as well as a mall, a train station, and a campus. In what follows, we summarize the authors' findings: (i) the social graph of all the

targeted events has social-network properties; (ii) the distributions of languages and mobile device vendors match the nature of the monitored crowds; and (iii) socially interconnected people tend to adopt mobile devices of the same vendor, and appear in the same time slot.

5.2.4 Traffic Characterization

Network management can benefit from knowing the properties of the Internet traffic that traverse the network. Such information can be used to efficiently deploy the hardware equipments, as well as to setup them in order to provide the best Quality of Service (QoS) to the users. This statement particularly holds for networks serving mobile devices, since such devices generate traffic with peculiar properties. In light of the rapid evolution of mobile devices, the characterization of their Internet traffic is crucial to provide network administrators the information they need for resource planning, deployment, and management.

We define as *traffic characterization* the analysis of the network traffic generated by mobile devices in order to infer its properties. We group the works that deal with traffic characterization into two sub-categories, according to the scope of the analysis:

- The works that study the network traffic of specific apps and/or mobile services. We survey nine works belonging to this category. Rao et al. in [127] study the Android and iOS native apps of two video streaming services, namely Netflix and YouTube. YouTube is targeted also in [124]. The Android apps of Facebook and Skype are considered in [128]. In [117], Wei et al. focus on 27 Android apps (19 free and 8 paid). In [144], the analysis covers over 1,000,000 unique Android apps. In [147], Chen et al. analyze 5560 malicious Android apps (from 177 malware families). In [159], Nayam et al. study 63 Android and 35 iOS free apps, all belonging to the "Health & Fitness" category. The work in [118] focuses on five interactive apps for both Android and iOS. In [168], Espada et al. present a framework for traffic characterization of Android apps, and choose Spotify as case study.

- The works that study the network traffic generated by a population of mobile devices. We can further divide such works into two subsets:

  - The works that compare mobile traffic with non-mobile one. We survey four works belonging to this subcategory. The work in [122] focuses on the network traffic of mobile devices when they are connected to home Wi-Fi networks, while the works in [119, 125, 126] carry out the same analysis for campus Wi-Fi networks.

– The works that only consider mobile traffic. We survey four works belonging to this subcategory. The works in [129, 171] target campus Wi-Fi networks, while the works in [120, 123, 148] leverage client-side measurements collected through logging apps.

Table 5.6: The surveyed works that deal with traffic characterization.

Year	Paper	Apps/Mobile Services	Mobile Devices	
			Non-mobile	Only Mobile
2010	Afanasyev et al. [119]		✔	
	Falaki et al. [120]			✔
	Maier et al. [122]		✔	
	Shepard et al. [123]			✔
2011	Finamore et al. [124]	✔		
	Gember et al. [125]		✔	
	Lee et al. [126]		✔	
	Rao et al. [127]	✔		
2012	Baghel et al. [128]	✔		
	Chen et al. [129]			✔
	Wei et al. [117]	✔		
2014	Lindorfer et al. [144]	✔		
2015	Chen et al. [147]	✔		
	Fukuda et al. [148]			✔
2016	Nayam et al. [159]	✔		
	Tadrous et al. [118]	✔		
2017	Espada et al. [168]	✔		
	Wei et al. [171]			✔

In Table 5.6, we provide a view of the classification described above. In what follows, we list the main properties of mobile traffic that stem from the works we survey:

- Compared to residential broadband traffic, the daily volume of traffic per mobile user is roughly one order of magnitude smaller [120].
- Mobile devices generate more downlink traffic than uplink one, showing a clear client-server network behavior [120, 117, 148, 118].
- At the network layer, IP flows of mobile devices have shorter duration, much higher number of packets, and much smaller packets, compared to IP flows of non-mobile devices [126].
- Most of the transport-layer traffic is carried over TCP [120, 125, 129], and more than half is encrypted [120]. Transfers within TCP connections are small in size [120, 125], causing a high overhead for lower-layer protocols, particularly when transport-layer encryption is in place [120].
- Most of the application-layer traffic is carried over HTTP or HTTPS [120, 125, 126, 129, 144, 147, 171]. Moreover, the analysis carried

out by Chen et al. in [129] shows that: (i) the adoption of HTTPS is increasing (a trend confirmed in [159, 171]); and (ii) Akamai and Google servers serve nearly 40% of the mobile traffic.

- Mobile devices contact a less diverse set of hosts compared to non-mobile devices [125, 129].
- Mobile devices experience a low loss rate on Wi-Fi networks [129]. On cellular networks, instead, there are high delays and losses, as well as low throughputs [120].
- An important fraction of the mobile traffic is due to video streaming [122, 125], mainly on the YouTube platform [124].
- Android apps typically do not encrypt their network traffic (simply relying on HTTP), connect to several different hosts, and a fraction of their network traffic is related to Google's services [117].
- A significant part of the network traffic generated by Android and iOS free apps is due to advertisement and tracking services [159].
- The Android apps of Netflix and YouTube tend to periodically buffer large portions of the video to be played, while their iOS counterparts tend to initially buffer a large amount of data, then periodically buffer small portion of the video to keep playback ongoing (although the YouTube app employs large-block buffering under favorable network conditions) [127]. Moreover, probably to deal with the TCP timeouts caused by the delays of cellular networks, the Netflix and Youtube iOS apps create a large number of TCP flows to provide a single video, thus causing an overhead that is not necessary when mobile devices are connected to Wi-Fi networks [127, 129].

5.2.5 Trajectory Estimation

In Section 5.2.1, we discussed the applications and privacy implications of device positioning. In this section, we present a different problem: to infer the trajectory followed by a mobile device in a geographical area, by analyzing the network traffic it generates. This type of traffic analysis, which we call *trajectory estimation*, can help study the interests and social habits of a mobile user (if we focus on a single individual), as well as aid traffic prediction along urban streets (if we aggregate the trajectories of several mobile users).

Musa and Eriksson in [131] present a system for passively tracking mobile devices by leveraging the Wi-Fi probe requests they periodically transmit. The idea is to employ a number of Wi-Fi monitors, which look for probe requests from mobile devices and report each detection to a central server,

where the detections of the same mobile device are turned into a spatio-temporal trajectory. To evaluate their system, the authors set up three deployments and leverage GPS ground truth to measure the accuracy of the inferred trajectories. The mean error is under 70 meters when the distance among the monitors is over 400 meters.

5.2.6 Usage Study

The habits of mobile users have significantly changed with the evolution of cellphones toward smartphones and tablets. First of all, the adoption of the touchscreen display has completely revolutionized the human-machine interaction. Moreover, the development of mobile operating systems supporting multitasking and third-party apps has enhanced the capabilities of mobile devices well beyond the requirements for communication activities. In this scenario, it is fundamental to understand how mobile users interact with their mobile devices in order to improve the usability of mobile OSes and apps, as well as to properly set up networks serving mobile devices. For instance, the knowledge of places where mobile devices are mostly used can drive the deployment of free Wi-Fi hotspots in order to reduce the traffic load on cellular networks.

We define as *usage study* the analysis of the network traffic of mobile devices in order to infer the usage habits of mobile users. The works we review in this section leverage network-side measurements [119, 122, 124, 125, 126, 171], as well as data collected within mobile devices [120, 130, 148, 151]. Overall, we identify three perspectives to study the usage habits of mobile users:

- The network. As an example, we can study when users are active (i.e., sending and receiving data) during the day, how long their periods of activity are, how much traffic they generate, and which are the most frequently used network interfaces (i.e., Wi-Fi or cellular).
- The apps and/or mobile services. As an example, we can study which are the most frequently used apps/services, and which is the traffic volume of a specific app.
- The geographical positions and mobility patterns. As an example, we can study where mobile devices are most frequently used, and where they generate most of their traffic.

In Table 5.7, we show the types of usage study carried out in the surveyed works that deal with this kind of traffic analysis. In what follows, we summarize their findings:

- The most frequently used apps are the ones related to multimedia content (e.g., YouTube, Spotify) and web browsers [120, 122, 125,

Table 5.7: The surveyed works that deal with usage study.

Year	Paper	Network	Apps/Mobile Services	Geography/Mobility
2010	Afanasyev et al. [119]	✔	✔	✔
	Falaki et al. [120]		✔	
	Maier et al. [122]		✔	
2011	Finamore et al. [124]		✔	
	Gember et al. [125]		✔	
	Lee et al. [126]	✔	✔	
2012	Ham et al. [130]	✔	✔	
2015	Fukuda et al. [148]	✔		
	Soikkeli et al. [151]	✔		✔
2017	Wei et al. [171]	✔		

126, 130, 148]. Social network and instant messaging apps are also popular [148].

- The predominance of cellular over Wi-Fi network traffic observed for mobile devices by Ham and Choi in 2012 [130] is gradually disappearing. As reported in [148], in 2015 more than half of mobile traffic is carried over Wi-Fi. In particular, mobile users tend to switch to Wi-Fi connectivity whenever a Wi-Fi access point is available [148].

- The usage of mobile devices is low at nighttime and high in daytime [119, 126, 130, 151].

- Cellular traffic peaks in commute times, while Wi-Fi traffic peaks in the evening [148]. Cellular traffic is lighter on weekends than weekdays, while Wi-Fi traffic follows the opposite trend [148].

- Mobile users tend to generate more network traffic when they are out of home, and when their devices have high battery level [151].

- The volume of traffic generated by the mobile users of a Wi-Fi network varies greatly, from less than 100 MB to several GBs, according to users' habits and needs [171].

- According to Finamore et al. in [124], YouTube users on mobile devices: (i) similarly to non-mobile users, they prefer short videos (40% of the watched videos are shorter than three minutes, and only 5% are longer than ten minutes); (ii) similarly to non-mobile users, they rarely change video resolution and, whenever they do that, it is to switch to a higher resolution (although full screen mode is not frequently used); and (iii) more frequently than non-mobile users, they early stop watching the video (within the first fifth of its duration for 60% of the videos).

5.2.7 User Action Identification

Most of the apps can leverage the Wi-Fi and cellular network interfaces of mobile devices to send and receive data. Since users perform several actions while interacting with apps, it is likely that most of such actions generate data transmissions. The network traffic trace of a given action typically follows a pattern that depends on the nature of the user-app interaction of that action. As a practical example, browsing a user's profile on Facebook will likely produce a different traffic pattern compared to posting a message on Twitter. These patterns can be used to recognize specific user actions related to a particular app of interest in generic network traces. Moreover, it is often possible to infer specific information about a given user action (e.g., the length of the message sent via an instant messaging app). We define as *user action identification* these types of traffic analysis.

The possibility to identify actions of mobile users can be useful in several scenarios:

- It is possible to profile the habits of a mobile user (e.g., checking emails in the morning, watching YouTube videos in the evening). The user's behavioral profile can be used to later recognize the presence of that user in a network. Moreover, profiles of thousands of mobile users can be aggregated in order to infer some information for marketing or intelligence purposes.

- It is possible to perform *user de-anonymization*. Suppose a national agency is trying to discover the identity of a dissident spreading anti-government propaganda on a social network. It is possible to monitor a suspect and detect when she posts messages via the social network mobile app. The inferred posting timestamps can be matched with the time of the messages on the dissident social profile in order to understand if the suspect is actually the dissident.

In Table 5.8, we show the app categories covered by the surveyed works that deal with user action identification [140, 142, 150, 45, 162]. Almost all the works target communication apps, which belong to the most privacy-sensitive app category. This category includes instant messaging apps (e.g., iMessage, KakaoTalk, WhatsApp) as well as email clients (e.g., Gmail, Yahoo Mail). Another sensitive category is social (e.g., Facebook, Twitter, Tumblr), which is targeted in [45, 162]. Apps related to multimedia contents (e.g., YouTube) are considered in [140, 162]. Moreover, Saltaformaggio et al. in [162] also focus on other categories of apps: dating (e.g., Tinder), health (e.g., HIV Atlas), maps (e.g., Yelp), news (e.g., CNN News), and shopping (e.g., Amazon). The works in [140, 45] cover productivity apps (e.g., Dropbox), and Watkins et al. in [140] also consider mobile games (e.g., Temple Run 2) and utility apps (e.g., ZArchiver).

Table 5.8: App categories that deal with user action identification.

Year	Paper	Covered App Categories										
		Communication	Dating	Gaming	Health	Maps	Media	News	Productivity	Shopping	Social	Utility
2013	Watkins et al. [140]			✔			✔		✔			✔
2014	Coull et al. [142]	✔										
2015	Park et al. [150]	✔										
2016	Conti et al. [45]	✔							✔		✔	
	Saltaformaggio et al. [162]	✔	✔		✔	✔	✔	✔		✔	✔	

Watkins et al. in [140] develop a framework that exploits the inter-packet time of responses to ICMP packets (i.e., pings) to infer the type of action that the target user is performing on her mobile device. In particular, the authors focus on three types of user action: (i) CPU intensive; (ii) I/O intensive; and (iii) non-CPU intensive. First of all, the authors check the feasibility of their approach for the Android and iOS platforms, showing that unfortunately their solution does not work for the latter because iOS does not use CPU throttling. Subsequently, they evaluate their framework using six Android apps, achieving a minimum 93% accuracy. Since the proposed solution exploits the timing of packets, it is not affected by traffic encryption.

Coull and Dyer in [142] target iMessage, Apple's instant messaging service, which is available as an app for iOS or a computer application for OS X. The proposed analysis leverages the sizes of the packets exchanged between the target user and Apple's servers, thus it works despite all iMessage communications are encrypted. The authors focus on five user actions: "start typing", "stop typing", "send text", "send attachment", and "read receipt". Assuming to have correctly inferred that the target mobile device is running iOS, all user actions can be classified with over 99% accuracy, except for the "read receipt" action that is often confused with the "start typing" action. The authors also aim to infer the language (among six languages: Chinese, English, French, German, Russian, and Spanish) and length of the exchanged messages. Assuming to have correctly identified an iMessage action on a mobile device running iOS, the language classification achieves more than 80% accuracy by considering the first 50 packets. Besides, the length classification achieves an average error of 6.27 characters for text messages, and an absolute error of less than 10 bytes for attachment transfers.

Park and Kim in [150] target KakaoTalk, an instant messaging service widely used in Korea. They consider eleven actions that a user can perform on the Android app (e.g., join a chat room, send a message, add a

friend). For each action, the proposed framework learns its traffic pattern as a sequence of packets. Such sequence is then used to recognize that specific action in unseen network traces. The proposed solution reaches 99.7% accuracy despite KakaoTalk traffic is encrypted.

Conti et al. in [45] present an identification framework which leverages the information available in IP and TCP headers (e.g., source and destination IP addresses) and therefore it works even if the network traffic is encrypted via SSL/TLS. However, the proposed approach does not work on an IPsec scenario, since it relies on (IP address, TCP port) pairs to separate traffic flows. The authors target seven popular Android apps (namely Dropbox, Evernote, Facebook, Gmail, Google+, Tumblr, and Twitter), reaching over 95% accuracy and precision for most of the actions, and outperforming websites fingerprinting algorithms by Liberatore and Levine [172] and Herrmann et al. [173].

Saltaformaggio et al. in [162] present NetScope, a user action identification system that can be deployed at Wi-Fi access points or other network equipments. Since it leverages IP headers/metadata, NetScope can be employed even if the network traffic is protected by IPsec. The authors evaluate their solution by considering 35 user actions from 22 apps across two platforms (i.e., Android and iOS) and eight app categories. The identification accuracy reaches average precision and recall of 78.04% and 76.04% respectively, performing better for Android devices rather than iOS ones.

5.2.8 User Fingerprinting

Mobile users interact actively with their devices, leveraging the nearly ubiquitous Internet connectivity and the capabilities of the apps available in the marketplaces. To each mobile user, it is possible to associate a set of preferred (i.e., most frequently used) apps and, for each of these apps, a set of preferred (i.e., most frequently executed) actions. Since most of the mobile apps are able to connect to the Internet, and many user actions within them trigger data transmission through the network, it becomes clear that the network traffic generated by a user is likely to present a fairly constant pattern across different devices, as well as across different networks. We define as *user fingerprinting* the attempt to exploit such pattern in order to recognize the network traffic belonging to a specific mobile user. This type of analysis can be applied to:

- Recognize the presence of a specific mobile user within a network. Once the network is identified, it is then possible to approximate the geographical position of that user with the location of the Wi-Fi hotspot or cellular station to which her mobile device is connected.
- Partition the network traces of a mobile traffic dataset by user. Once the transmissions related to a specific mobile user have been separated

and grouped together, it is then possible to apply other types of traffic analysis targeting that user.

In this section, we review the works that deal with mobile user fingerprinting. Vanrykel et al. in [164] investigate how mobile unencrypted traffic can be exploited for user surveillance. The authors develop a framework to automatically execute apps, collect their network traffic, inspect the HTTP data, and identify the sensitive identifiers that are transmitted in clear. Moreover, the authors present a graph building technique that exploits such identifiers to extract the network traces generated by a specific mobile user from a traffic dataset. The analysis of 1260 Android apps (from 42 app categories) shows that the proposed solution can link 57% of a mobile user's unencrypted network traffic. In addition, the authors observe the limited effectiveness of ad-blocking apps in preventing the leakage of sensitive identifiers.

Verde et al. in [146] present a system being able to accurately infer when a target user is connected to a given network and her IP address, even though she is hidden behind NAT among thousands of other users. To achieve this objective, a machine learning classifier is trained with the NetFlow records of the target user's traffic, and then employed to analyze the NetFlow records of a given network in order to detect the presence of the target user within it. The system is evaluated as follows:

- Cross-validation is applied to the NetFlow records of the traffic generated by 26 different mobile users connecting to the Internet through the same Wi-Fi access point. The best performing implementation of the classifier (which is based on random forest) reaches 95% true-positive rate, 7% false-positive rate, 0.95 precision, 0.93 recall, and 0.94 F-measure.

- To profile them, five mobile users are induced to connect to a Wi-Fi access point under the control of the authors, who then try to detect the presence of such users within a real-world large metropolitan Wi-Fi network. The implementation of the classifier based on random forest (the best performing) reaches over 90% true-positive rate, less than 8% false-positive rate, and over 0.9 precision and recall.

It is worth to notice that the proposed solution is encryption-agnostic, since NetFlow records can be extracted even from encrypted traffic.

5.2.9 Website Fingerprinting

The Internet has a central role in people's everyday life, and surfing the Web has become a common task that can be performed from desktop computers, as well as in mobility using laptops and smartphones/tablets, thanks to the increasing deployment of cellular and Wi-Fi networks. From a privacy point of view, the set of websites frequently visited by a user is a sensitive

information, since it can disclose her interests, social habits, religious belief, sexual preference, and political orientation.

In the field of traffic analysis, *website fingerprinting* generally indicates the attempt to infer the website or even webpage visited by a user surfing the Internet, by analyzing the network traffic generated by her web browser. This type of analysis has been extensively treated in the domain of personal computers, where machine learning techniques have been proved to be very effective [172, 173, 174]. Since we focus on mobile devices, in this section we survey the works that target users navigating through the web browser of their mobile devices.

Spreitzer et al. in [163] develop an Android app being able to capture the data-usage statistics of the browser app, and leverage them to fingerprint the webpages visited by the user of the mobile device on which that app is installed. This solution is not affected by encryption, since it only requires to know the amount of data transmitted and received by the browser app (which is easily obtainable in Android). The proposed app is evaluated on a set of 500 possible pages that the user can visit. Overall, 97% of 2,500 page visits are correctly inferred. The authors also evaluate their fingerprinting app when the network traffic is protected by Tor (in particular, by using the Orbot proxy and the Orweb browser). Out of a set of 100 possible pages that the user can visit, 95% of 500 page visits are correctly inferred.

5.3 Summary

In this chapter, we surveyed the state of the art of the work for analyzing the network traffic generated by mobile devices. In particular, we are the first that surveyed the works in which the mobile traffic is captured from alternative sources to cellular networks: Wi-Fi monitors and access points; wired networks; logging apps installed on mobile devices; and networks of mobile devices simulated via software. Moreover, we observed that the most frequently used approach to capture mobile traffic is logging at either wired networks or mobile devices themselves. We provide a systematic classification of the state of the art according to the goal of the analysis that targets the network traffic of mobile devices. In particular, we realized that most of the work focus on studying the features of the network traffic generated by mobile devices, especially related to Human-Computer Interaction, such as user action identification, usage study, and Personal Identifiable Information (PII) leakage. While a lot of work has been done on such goals, promising topics, such as user fingerprinting and sociological inference, still offer much space for further investigation. Regarding the feasibility of the surveyed analyses in case of traffic encryption, we observed that 36 out of 56 work proposed methods that are able to carry out the analysis on traffic encrypted with SSL/TLS, while only 19 out of 36 can handle network traffic

encrypted with IPsec. As in the traditional network traffic domain, the feasibility of the analysis on the encrypted traffic of mobile devices is bounded to the network layer in which the traffic has been captured.

Chapter 6

Network Traffic Management via SDN

Software Defined Networking (SDN) is an emerging networking architecture which advocates better decoupling between network control (*control plane*) and data forwarding functionalities (*data plane*). In SDN [175], control plane manages a collection of OpenFlow-enabled switches in data plane which is responsible for data processing and forwarding. The functional separation makes the management of SDN network more flexible, which allows the users and administrators to design their own software routine to dynamically manage the network. Network management is becoming more and more vexing, since network administrators perform increasingly sophisticated network management tasks. Moreover, if the configurations [17] are based on some advanced commands, it is error prone for the users who rely on the Graphical User Interface (GUI) to manage and operate the network. The two main causes of errors are the dynamically changing network state and low-level network configuration. The state-of-the-art methods for network management do not allow users to configure network policies automatically reacting to low-level dynamic network activities. Compared with the traditional network, SDN provides several benefits for improving network management [176], thanks to its functional separation in different layers. The first benefit introduced by a SDN architecture is that instead of using set of abstract and error-prone commands, SDN makes it feasible to manipulate the network devices by easily designing, distributing innovative flow handling and network control algorithms. Second, centralized SDN allows to configure all the network devices once if there are changes of network behavior. Third, SDN supports the abstraction of network configuration relying on controllers (e.g., Floodlight [177], NOX/POX [178], OpenDaylight (ODL) [179]), which get global knowledge of network state and support au-

tomatically reacting to dynamic network events. However, there still exist a stark paucity of programming SDN security applications. In this chapter, we present *GolfEngine*, a novel SDN management architecture, providing users with a tool to build their security applications for SDN network. Our proposal offers an intuitive GUI for managing and monitoring the state of the whole system. Indeed, GolfEngine users will be able to define rules and design their security application even without any programming knowledge. Designing an application, GolfEngine also helps users to check policy conflicts. Furthermore, our GolfEngine provides the mechanism to check data storage redundancy, which improves the network security application efficiency.

Chapter Organization. The rest of the chapter is organized as follows. In Section 6.1, we briefly survey the state of the art of SDN network management. We give a detailed description of our SDN management architecture, *GolfEngine* in Section 6.2. In order to illustrate the work flow of GolfEngine concretely, a use case *Firewall* is presented in Section 6.3. In Section 6.4, we describe simulation and evaluation results. We conclude the chapter in Section 6.5.

6.1 Related Work

Several works have contributed a lot to improve the security management of SDN network. M.K. Shin et al. in [180] discussed a SDN reference architecture and APIs for networking environment. In order to assure network security and correctness, the work in [181] helped the controller developers make their programming efficient through measuring OF switches performance. They pay attention on analyzing control plane processing times and flow table update rate. Instead, our GolfEngine is designed on top of control plane to solve the SDN network management. Shin et al. in [182] proposed FRESCO, an OF security application development framework designed to facilitate the rapid design. Such framework allowed to design modules to perform complex OF-enabled network security services (e.g., detection and mitigation modules), which can be built from smaller sharable libraries of security functions. Nevertheless, FRESCO design scale was at module level and it required users to have a medium level skill with Python programming language in order to develop application modules. Moreover, FRESCO was based on the NOX controller. Instead, GolfEnigne is based on ODL controller and it is system-level for designing security application with free conflict. Authors in [176, 183] designed and implemented PROCERA, an event-driven framework for high-level reactive SDN network control. It simplified network operations and management from three aspects: enabling changing network conditions and state frequently; providing supports for high-level

language for low-level network configuration and policy; and supplying better visibility and control over tasks for performing network anomalies. M. Monaco et al. in [184] introduced YANC, a controller platform for SDN. They considered the network configuration and state as a file system, which allows users and system applications to interact through standard file I/O, and to easily leverage tools available on the host operating system. They completed a demonstration with a simple static flow pusher application. In literature [185], authors proposed an extended SDN architecture that enables fast customized packet handling. They focused on the data plane processing according to applications deployment. In [186], they introduced FortNOX, a SDN application extension that provides authorization and security enforcement based on NOX controller. FortNOX allowed NOX to check flow rule contradictions in real-time. Moreover, they demonstrated the performance and efficiency of FortNOX through a prototype implementation. Compared with FortNOX, GolfEngine not only provides the mechanism of avoid communication redundancy, but also it improves the efficiency of process policy contradictions. In the work [187, 188, 189], authors talked about firewall policy based on traditional network. They gave definition of rule relation and kind of rule conflicts and also the algorithm to check the conflicts. Moreover, in order to check the conflict, they adopt the mechanism of designing the policy tree to compare with every new rule. Authors in [190] presented FireWell, a firewall policy analyzer based on Alloy to detect conflicts of policies in SDN network. Firewall read, validate, detect and prevent policy conflicts from SDN Floodlight controller in Firewall application. They focused on evaluating and optimizing the performance of the conflict detection. Besides, they mapped firewall policies into Alloy models to translate abstract policies. Different from them, a use case of GolfEngine (e.g., Firewall) redesigns the rule format, redefine relation between two rules, and redevelop algorithm for conflict checking. GolfEngine amazingly improves the efficiency of conflict detection comparing to the work [187].

6.2 GolfEngine Architecture

We name our network management architecture for SDN *GolfEngine*, because we expect our proposal to achieve our target as the way playing a golf into a hole. In this section, first we give a brief introduction of GolfEngine architecture layers. Then we illustrate each layer in detail.

6.2.1 Architecture Introduction

As shown in Figure 6.1, the left part shows the normal SDN architecture, consisting of *Data Plane*, *Control Plane* and *Application Layer*. Normal SDN architecture offers an application layer to develop, test, and run application to control the network security operation. Besides, the OF proto-

col enables switches to communicate with the controller and the controller to manage the switches. The right part of Figure 6.1 depicts GolfEngine architecture. The red, green, yellow and blue part represent *GolfEngine Application*, *User Interface*, *Database*, and *Runtime Manager*, respectively.

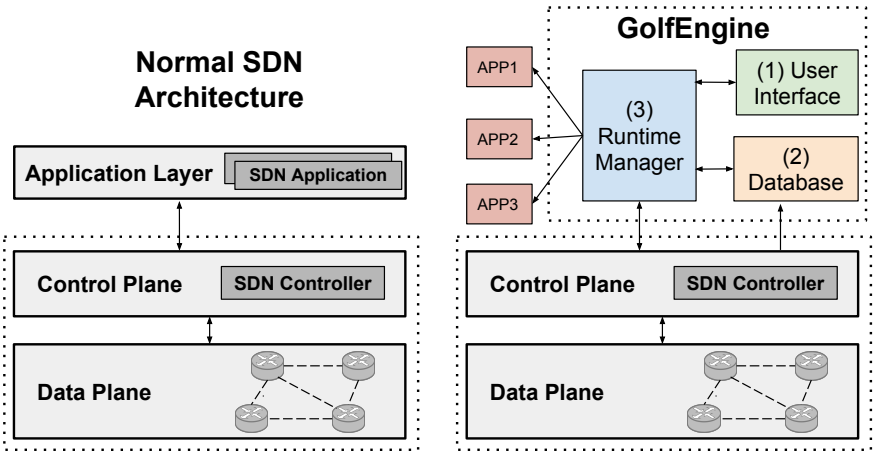


Figure 6.1: Structure comparison between GolfEngine and normal SDN.

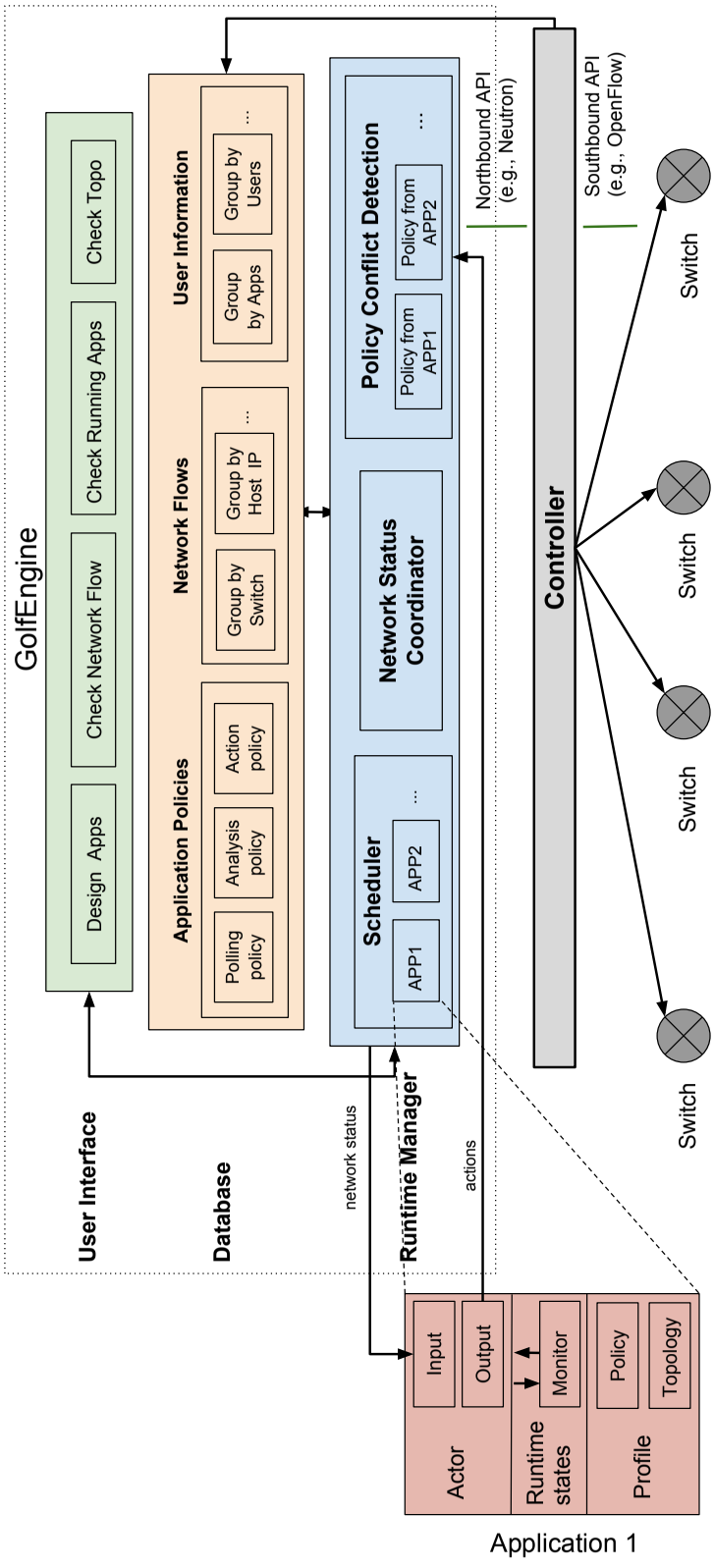


Figure 6.2: GolfEngine architecture.

In Figure 6.2, we show the structure of GolfEngine layers in a more detailed schema. GolfEngine system works on top of SDN controller using the Northbound API, which is like a luxuriant application layer for the original SDN prototype. GolfEngine consists of the three aforementioned layers, *User Interface*, *Database*, and *Runtime Manager*. GolfEngine not only allows users to design their own proprietary *GolfEngine Application* via *User Interface*, but also *Runtime Manager* supports users to manage the network dynamically and design policy conflict-free *GolfEngine Applications*.

In the following sections 6.2.2, 6.2.3 and 6.2.4, we give a description of these three layers of GolfEngine.

6.2.2 User Interface

User Interface is the first layer of GolfEngine. It provides convenient entrance for users to design applications and check on-time network statistic and running applications, and network topology. More in detail, *User Interface* implements four functions:

- **Design applications:** It is the main function of User Interface layer. As a proof of concept, we implement three kind of security applications for users to design: Firewall, HoneyPot, and DDoS Mitigate.
- **Visualization of the network topology:** Users have under their control the status of the whole virtual network thanks to an intuitive graphical representation of its topology.
- **Get network flow:** Network administrators can check network flows properties (e.g., packet number, bytes number, bandwidth), and GUI can allows users to visually observe such data in a given time interval.
- **Check running applications:** This function allows users to check the applications which is running on GolfEngine platform.

In summary, the GUI design significantly simplifies the management, while minimizing network vulnerability of misconfiguration.

6.2.3 Database

This layer of GolfEngine architecture assures applications of a good communication between users and SDN controller. In our prototype implementation of GolfEngine, we utilize MySQL as database managing system to store the data for the implementation of GolfEngine. *Database* layer is comprised of three stand-alone types of data:

- **Application Policies Database** stores elements that users may need to design their own applications. We pre-define several useful policies according to the users' requirements of designing applications. We

store into the database three types of application policies: (i) polling policy (e.g., polling the number of packets through a specific switch), (ii) analysis policy (e.g., comparing the value given in polling policy with a specific threshold) and (iii) action policy (e.g., drop, forward).

- **Network Flows Database** stores all the networking statistics, which are polled by controller from switches. Such database is updated at a specified time interval in order to guarantee the freshness of collected network data. GolfEngine applications monitor such data in order to promptly react to network events with on-time. Unfortunately, monitoring network behavior at application level increases the communication load between the controller and switches. In order to cope with this problem and avoid data storage redundancy and communication redundancy, we adopt a GolfEngine system-level mechanism *Network Status Coordinator* (NSC) (see detail in Section 6.2.7) as a delegate to communicate with applications.

- **User Information Database** is used to log users' operations done on GolfEngine platform. GolfEngine logs in this database the behavior of each user and every relevant information users set during the applications designing process. As a practical example, a user who wants to design an application on GolfEngine to monitor the network, must register with an unique username. After the system administrators grant her the permission, she can select the desired application from our application list (e.g., Firewall). Thus, after choosing the application, she is allowed to set up some parameters according to her requirements for managing network. The corresponding information for her application requirements and parameters are recorded into User Information Database. As a future work, we can use machine learning techniques to analyze users' behavior and information.

6.2.4 Runtime Manager

Runtime Manager is the core layer of GolfEngine. The main contribution of GolfEngine is implemented by *Runtime Manager* layer. Runtime Manager layer is in charge of tracing, monitoring, reacting dynamically against critical network events (e.g., DDoS). It consists of three components:

- **Scheduler:** Similar to an operating system, in a SDN system, applications concurrently run sharing computational and networking resources. Therefore, the Runtime Manager relies on a scheduler to manage all the running applications in order to strictly regulate their access to resources, avoiding deadlocks or resource starvation.

- **Network Status Coordinator:** It is a system-level application that polls network data from switches and stores into database to avoid

communication redundancy. See detail in Section 6.2.7. In case two different applications poll different data from switches independently, it will not occur communication redundancy. However, if these two applications poll the same data, they will request the controller to poll the same data from switches twice. Such overlapped communication between the controller and same switches for same data wastes both network resources and time. Introducing NSC mechanism, applications first request NSC for the data. If such requested data have been already cached in Network Flows Database by the NSC due to another recent request (i.e., cache hit), NSC sends to applications directly. Or else (i.e., cache miss) NSC polls the requested data from switches and stores it into Network Flows Database.

- **Policy Conflict Detection:** It is a mechanism that handles flow rules (responding to/interact with networking behaviors) from outputs of applications and the action policy (see Section 6.2.3). PCD is responsible for detecting the anomalous conflict among action policies. A policy conflict is defined as a phenomenon, in which more different filtering rules match to the same packet. This phenomenon contains conflicts that cause some rules to be always subverted by other rules, or warnings for potential conflicts between related rules. First, PCD checks the relation of policy based on the fields of the policy rules. Then, it detects the conflict anomaly according to the relation, priority level and action. We describe in detail the PCD algorithm in Section 6.3.4.

In what follows, we describe the components of Runtime Manager layer in detail in Sections 6.2.5, 6.2.6 and 6.2.7.

6.2.5 GolfEngine Applications

Besides designing the architecture for SDN, we also design our own SDN applications for GolfEngine, especially network security applications (e.g., Firewall, HoneyPot, DDoS Mitigate). As depicted by Figure 6.2, each GolfEngine application is composed of three parts:

- **Profile:** This part consists of a pair made of a basic policy and a network topology.
- **Runtime States:** It monitors at runtime the network status based on the applications' rules.
- **Actor:** It is sub-divided into input and output. An application gets the network status from input APIs. Applying basic policy, the application generates actions and sends command to system's PCD component through output APIs.

We give an example to illustrate the work-flow of Firewall application designed with GolfEngine. Regarding the *profile* part, the topology for Firewall is topology of a network domain. There are three policies for Firewall: *Polling policy* is polling the number of packets transmitted through Switch 1 every 5 seconds. *Analysis policy* is whether this number is bigger than the given threshold 200. And *action policy* is blocking the traffic if the number is more than 200. In the *runtime states* part, Firewall application refreshes the network status (traffic through Switch 1) at a specific time interval (e.g., 5 seconds) to get the latest data. For *actor* part, the input is the network statistic (e.g., the number of packets transmitted through Switch 1) that Firewall application needs to make its decision, while the output is the action (e.g., block the traffic through Switch 1). When designing GolfEngine applications, we provide different APIs for users to design their customized applications.

6.2.6 Policy Conflict Detection (PCD)

Policy conflict is defined as a phenomenon that more than one different filtering rules matching the same packet. This includes clear conflicts that cause some rules to be always suppressed by other rules, or warnings for potential conflicts between related rules.

PCD is responsible for detecting the anomalous conflict among the policies. As shown in Figure 6.3, we provide the work-flow of PCD in GolfEngine.

First, we provide users the interface to create policies, according to a rule encoded in a specific format (for example the rule for Firewall application is shown in Figure 6.5). Subsequently, PCD optimizes the rules, on the basis of each field value provided by users in the previous step. In case of conflict-free scenario, PCD simply stores this rule into Application Policies Database. For the time being, GolfEngine checks the rules stored in the database that are still not yet executed but valid, and sends to the SDN controller. In case rule has conflict with the existing rules, PCD returns users an alert to delete and add a new one instead of modifying. We opt for this designing choice because of the results reported in [181], where the authors show that modifying a rule is slower than deleting it and adding an updated rule.

6.2.7 Network Statistic Coordinator (NSC)

In GolfEngine, we include a component NSC to cope with communication redundancy issue. Different GolfEngine applications may poll the same network information from the controller independently, thus communications between the controller and switches are redundant. However, when different applications poll different network data from the controller independently,

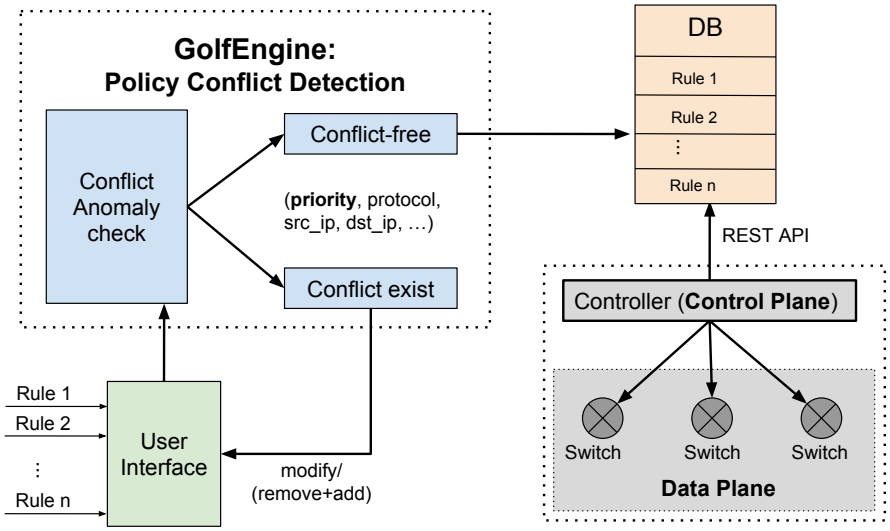


Figure 6.3: Policy conflict detection principle.

there does not exist communication redundancy. In what follows, we present an example for redundant communication problem. Given an *Application A*1 that wants to get the bandwidth information between *Switch X*and *Switch Y*, thus *Application A*1 will poll bandwidth information through the controller. Without any coordinator in place, whether *Application A*2 also wants to get the same information that *Application A*1 previously polled, *Application A*2 will ask the controller such information again. Such overlapped communication between the controller and same switches for same data wastes both network resources and time. Besides, applications waste a not negligible amount of time waiting of controller's reply and to get the desired data.

NSC is a system level component, which is used to store the data polled from switches and hosts into database to avoid communication redundancy. Here we present again the same scenario of the previous example but with the NSC in place. Given an *Application A*1 tries to poll the bandwidth information between *Switch X*and *Switch Y*, it sends a request to NSC first. NSC checks whether Network Flows Database has already cached this data. If cached, NSC forwards the data to *Application A*1 directly. If there is no corresponding data, NSC polls this data from the controller and stores into Network Flows Database. Later if *Application A*2 wants the same data with *Application A*1, it can directly get the data from the local database, which saves a lot of time compared with the previous scenario. It is worth noting that with our NSC in place not only can avoid resource wasting, but it can also save time in retrieving the network data.

6.3 A Use Case for GolfEngine Security Application: A Firewall

In order to better comprehend GolfEngine operating principle, we present a use case of a *Firewall* application to concretely illustrate step by step the development of an application with GolfEngine. Firewalls are core elements in the field of network security. Based on a specific security policy, firewalls control network packets direction, whether the packets are allowed to traverse across the boundaries of a secured network. Firewall policy is interpreted by ODL controller to define actions and rules to process the network traffic. Moreover, the optimization of policies plays a crucial role in making a firewall stronger against DoS and DDoS attacks. For the sake of security and efficiency of a network, firewall rules have to be conflict-free. We can find some work talking about firewall conflicts in traditional network [191] and SDN network [190].

The firewall security policy consists of a list of filtering firewall rules executed by order. In the following, we describe our rule format for Firewall of GolfEngine in Section 6.3.1, the mechanism of detecting policy conflict in Section 6.3.2, 6.3.3 and 6.3.4.

6.3.1 Firewall Rule Format

The firewall security policy consists of a list of filtering rules executed by order. These policies are used to define constraints, limitations and authorization on the network data and communication. The filtering rule decides the action for the packets which match the rule. The filtering rule is composed of several related fields (e.g., protocol type, source IP address, destination IP address, source port and destination port, and an action) with packets. For the sake of compatibility, Firewall rule format should not only conform to traditional firewalls standards [191], but also support OF-enabled switches. As described in OpenFlow white paper [192], an OF-enabled switch has one or more flow tables installed by the controller. A flow table is a set of flow entries, while a flow entry consists of several filter fields, counters, and a set of instructions (see Figure 6.4). These filter fields are corresponding to the fields described in tradition Firewall [193], while the instructions in the flow entry correspond to the actions of Firewall.

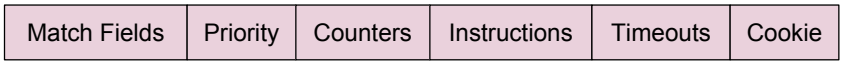


Figure 6.4: The format of flow entry on OF switches.

In GolfEngine architecture, we design a classic and popular security application Firewall, whose rule format is shown in the first row of Figure 6.5.

- **priority:** The value in this field represents priority level of this rule executed by the controller. The value is between 0 and 65535. The bigger value, the higher priority to be executed. The default value is hashing the timestamps when users create rules. Priority field is closely associated with the conflict of policy (see detail in Section 6.3.3). Moreover, PCD not only checks the rules with those who have priority higher than n, but also checks with those who have priority lower than n.
- **protocol:** The value can be TCP, UDP, IP and so on.
- **src_ip; dst_ip:** They represent source and destination IP address, respectively. The value can be a specific IP address (e.g., 10.0.0.1) or even whole network (i.e., *.*.*.*).
- **src_port; dst_port:** They represent the source and destination port, whose value can be a specific value (e.g., 80, 53) or a general value (e.g., any).
- **valid_time:** The value of this field indicates the length of time during which a rule will be valid on the controller.
- **action:** The policy actions in Firewall are either to accept or to deny. In our use case application, the default policy action is set to "deny".

In order to develop a Firewall application, it is necessary to model Firewall rule according to the function of each field. In the following sections, we define the relation of two rules, the conflict classification and detect conflict based on this rule model. For these fields, five of them are used to define the relation between two rules. Subsequently, the relation, the field of priority and action are leveraged together by PCD to detect any conflict (shown in Figure 6.5). The algorithm used by PCD for conflict detection is described in detail in Section 6.3.4.

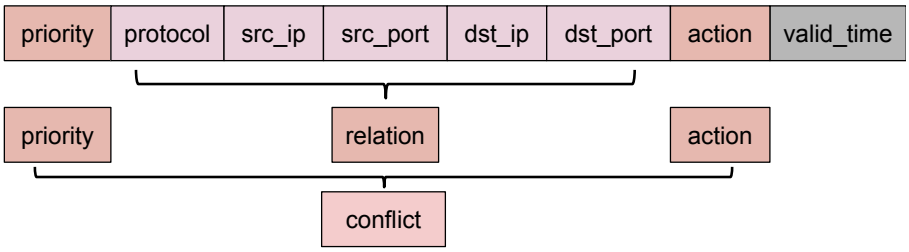


Figure 6.5: The format of Firewall rule.

6.3.2 Rule Relation

The five following fields (i.e., *protocol*, *src_ip*, *dst_ip*, *src_port* and *dst_port*) are included into rule relations. We classify the relations into five types:

- **SAME ($R_x\_value = R_y\_value$):** If two rules keep a SAME relation, it means that these five fields values in R_x exactly equal to the corresponding fields in R_y.
- **SUB ($R_x\_value \subset R_y\_value$):** If two rules keep a SUB relation, it means there exists at least one field value in R_x is a proper subset of the related field in R_y, while the remaining fields in R_x are exactly equal to R_y.
- **SUPER ($R_x\_value \supset R_y\_value$):** If two rules keep a SUPER relation, it means there exists at least one field value in R_x is a proper superset of the field in R_y, while the rest fields in R_x are exactly equal to R_y.
- **SUBSUPER ($R_x\_value \subsetneq R_y\_value$):** If two rules keep a SUBSUPER relation, it means that some fields in R_x are subsets or equal to the corresponding fields in R_y, and the rest fields in R_x are supersets of the corresponding fields in R_y.
- **EMPTY ($R_x\_value \emptyset R_y\_value$):** Such relation means that at least one field in R_x is not equal to, proper subset, and superset of the corresponding field in R_y.

6.3.3 Policy Conflict Classification

The ordering of rules in a security policy is crucial in designing the firewall policy. This is because the firewall packet filtering process is performed by sequentially matching the packet against filtering rules until a match is found. Therefore, if the ordering of conflicting rules is not processed carefully, it will result in producing an improper security policy and action. Here we give a clear definition of these four kinds of conflicts, especially the value of action, the ordering of two rules.

- **SHADOWED:** R_x is shadowed by R_y iff:
  - (i) $R_x\_priority < R_y\_priority$; (ii) $R_x\_action \neq R_y\_action$; and (iii) $R_x\_value \subseteq R_y\_value$.

  The third constraint means that R_x and R_y keep a SAME or SUB relation. We consider SHADOWED as an error. One reason is because the shadowed rule will never be executed, which means a waste of time and space resources. Another reason is that SHADOWED conflict might block network traffic that should be permitted and vice versa.
- **GENERALIZATION:** R_x is a generalization of R_y iff:
  - (i) $R_x\_priority < R_y\_priority$; (ii) $R_x\_action \neq R_y\_action$; and (iii) $R_x \supseteq R_y$.

The third constraint represents that R_x and R_y keep a SAME or SUPER relation. It is just considered as a warning.

- **CORRELATION:** R_x and R_y have a correlation conflict in one of the following two conditions:

  1. (i) $R_x\_priority > R_y\_priority$; (iii) $R_x\_action \neq R_y\_action$; and (iii)$R_x\_value \subseteq\supseteq R_y\_value$.
  2. (i) $R_x\_priority < R_y\_priority$; (ii) $R_x\_action \neq R_y\_action$; and (iii) $R_x\_value \subseteq\supseteq R_y\_value$.

  The third constraint is explained in Section 6.3.2. Moreover, CORRELATION is considered as a warning.

- **REDUNDANT:** R_x is redundant to R_y if these two rules satisfy one of the following two conditions:

  1. (i) $R_x\_priority > R_y\_priority$; (ii) $R_x\_action = R_y\_action$; and (iii) $R_x\_value \subseteq R_y\_value$.
  2. (i) $R_x\_priority < R_y\_priority$; (ii) $R_x\_action = R_y\_action$; and (iii) $R_x\_value \subseteq R_y\_value$.

  For the second condition in REDUNDANT, we also consider it as a redundant conflict, while F. A. Maldonado-Lopez et al. in [190] ignore this situation. For example, R_x is redundant to R_y. Even if they exchange the ordering between R_x and R_y (it is priority in our GolfEngine), they are still redundant to each other. REDUNDANT is considered as an error. For the first condition, even if R_x is removed from the Firewall application, it makes no difference for packets which match this rule. For the second condition, R_x will never be executed. Therefore, for both of these two conditions, redundant rules occupy space of the flow table, increasing the searching time as well.

  It is worth to notice that, for these four definitions, we are different from the work [190] and [194], since we take the field of priority and different scenarios into consideration.

6.3.4 Algorithms for Policy Conflict Detecting

Policy Conflict Detection (PCD) is a component to assure our GolfEngine system to be conflict-free. In this section, we describe our three main algorithms for detecting conflict. In what follows, we briefly describe the process of our PCD algorithms:

Insert a new rule

The aim of Algorithm 1 is to find a proper position for each rule according to its priority. Once it is set in the proper position, we can leverage the following two algorithms on every two rules among the rules set.

Algorithm 1 Insert a new rule

```
1: insertRule (rule_a):
2: position_found = 0
3: for i in Rules.size do
4:     if rule_a.priority > Rules[i].priority then
5:         Rules.insert (i, rule_a)
6:         position_found = i
7:     end if
8: end for
9: if position_found = 0 then
10:    Rules.insert (rule_a)
11: end if
```

Discover the relation between two rules

In order to determine the exact type of conflict, we check the relation between two rules. However, it is not necessary to identify all the relations described in Section 6.3.2. Because if two rules have EMPTY relation, they are conflict-free. In Algorithm 2, we discover the relations leading to conflicts.

Decide the classification of conflicts

Algorithm 3 is to decide the classification of conflict between two rules based on the definition for conflicts category in Section 6.3.3. After inserting a rule into a Firewall policy, we can be in two cases. In the first case, a conflict warning has been reported (i.e., GENERALIZATION, CORRELATED) between *rule_a* and the rule sets already deployed in Firewall, then *rule_a* is allowed to be inserted into the Firewall application as a "good" rule. In the second case, a conflict error has been reported (i.e., REDUNDANT, SHADOWED), thus the network manager can decide to insert *rule_a* into Firewall or not according to its priority. For example, if *rule_a* is contradictory with a lower priority rule, the one who has conflict with *rule_a* is removed from the Firewall application. However, if *rule_a* is conflicted with a higher priority rule, *rule_a* will be filtered by PCD. The algorithms by Ehab S. Al-Shaer et al. [187] and Ferney A. Maldonado-Lopez et al. [190] adopt Policy Tree to check the conflict. Compared with their algorithms, our algorithms not only ensure the correctness and the usability of our GolfEngine, but also

Algorithm 2 Discover relation between two rules

```
1: discoverRelation (rule_a, field_a, rule_b, field_b, relation):
2: if field_a != ACTION then
3:     if field_a = field_b then
4:         if relation = null then
5:             relation = SAME
6:         end if
7:         discoverRelation (rule_a, field_a.next, rule_b, field_b.next,
   relation)
8:     end if
9:     if field_a ⊂ field_b then
10:        if relation = SUPER then
11:            discoverRelation (rule_a, field_a.next, rule_b, field_b.next,
   SUBSUPER)
12:        end if
13:        if relation = SAME or SUB or null then
14:            discoverRelation (rule_a, field_a.next, rule_b, field_b.next,
   SUB)
15:        end if
16:    end if
17:    if field_a ⊃ field_b then
18:        if relation = SUB then
19:            discoverRelation (rule_a, field_a.next, rule_b, field_b.next,
   SUBSUPER)
20:        end if
21:        if relation = SAME or SUPER or null then
22:            discoverRelation (rule_a, field_a.next, rule_b, field_b.next,
   SUPER)
23:        end if
24:    end if
25: else
26:    decideClassification (rule_a, rule_b, relation)
27: end if
```

they reduce the computation complexity, and improve the efficiency and performance.

6.4 Simulation and Evaluation

In the prototype implementation, we build GolfEngine framework on OpenDayLight (ODL) that is a SDN controller written in Java. Besides, ODL supports OF protocol which allows the communication between controller and OF switches. However, we underline that our framework can be easily

Algorithm 3 Decide the classification of conflicts

```
1: decideClassification (rule_a, rule_b, relation):
2: if rule_a.priority < rule_b.priority then
3:     if rule_a.action = rule_b.action then
4:         if relation = SUB or relation = SAME then
5:             Report (rule_a, REDUNDANT, rule_b)
6:         end if
7:     else
8:         if relation = SUB or relation = SAME then
9:             Report (rule_a, SHADOWED, rule_b)
10:        end if
11:        if relation = SUBSUPER then
12:            Report (rule_a, CORRELATED, rule_b)
13:        end if
14:        if relation = SUPER or relation = SAME then
15:            Report (rule_a, GENERALIZATION, rule_b)
16:        end if
17:    end if
18: end if
19: if rule_a.priority > rule_b.priority then
20:    if rule_a.action = rule_b.action then
21:        if relation = SUB or relation = SAME then
22:            Report (rule_a, REDUNDANT, rule_b)
23:        end if
24:    else
25:        if relation = SUBSUPER then
26:            Report (rule_a, CORRELATED, rule_b)
27:        end if
28:    end if
29: end if
```

adapted to work on other SDN controllers. Our simulation implementation is done via Mininet [195].

In designing GolfEngine, we have to take into account not only its robustness and correctness, but also to ensure its efficiency. In what follows, we first show the efficiency of PCD in Section 6.4.1, and we then assess the optimal number of sending rules to controller for installing on the switches in Section 6.4.2.

6.4.1 Execution Efficiency of PCD

In our experiment, for each scenario we run PCD algorithm 1000 times to get the convincing plot. In Figure 6.6, we illustrate the results through a series of

cumulative distribution functions representing the processing time spent (in a log scale) to discover rule contradictions. We can notice that the processing time of checking conflicts increases almost linearly with the number of rules increasing. For each scenario (e.g., 1 rule, 10 rules), the values on Y-axis represents the probability to observe a processing time smaller than the corresponding value on X-axis. We can observe that each scenario exists an upper bound to the processing time (i.e., the corresponding value with $p = 1.0$). For example, in the scenario with 500 flow rules, the probability of processing time less than $2ms$ is 0.45, moreover, the probability of processing time smaller than $12.5ms$ is 1.0.

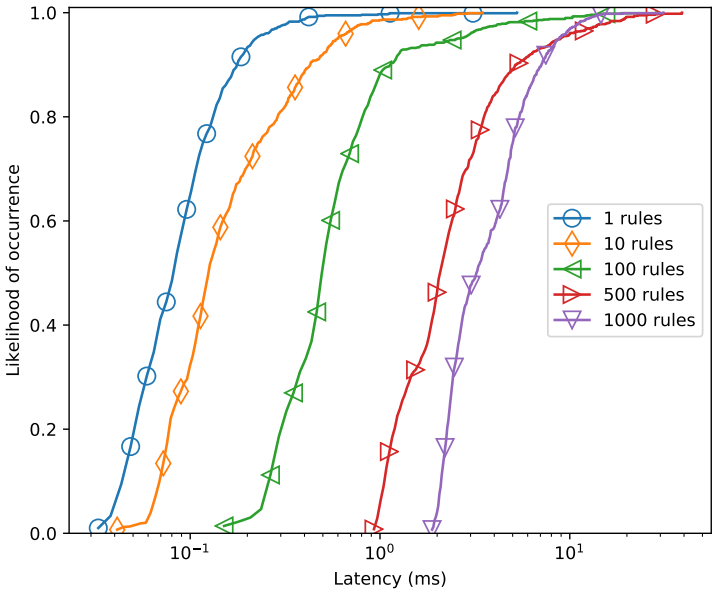


Figure 6.6: The CDF plots illustrate PCD performance (processing time) in conducting conflicts in scenarios (1, 10, 100, 500 and 1000 flow rules).

We adapt the same simulation setup in terms of the number of switches and hosts, in order to compare with the algorithms in [189, 190], . The results prove that PCD algorithm performs much more better than that in [189, 190] with respect to execution time for discovering and processing conflicts. In particular, it costs almost $100ms$ to detecting conflicts even in the best condition in [189], while it only costs more or less $1ms$ (range from $0.2ms$ to $12ms$). Moreover, for the last scenario (i.e., 1000 rules), compared with the computational delay of FortNox in [186], our PCD algorithm performs better. Moreover, it exists a corresponding value on X-axis in which the probability reaches 1.0, while FortNox in [186] such value does not exist.

6.4.2 Sending and Installing Rules on Switches

In our evaluation, we also take into account the time needed to send and install rules on switches. In the literature, SDN frameworks underestimate such important aspect [190, 189]. As an example, FireWell [190] is a type of firewall application designed for SDN network. FireWell communicates with the controller directly when it detects rules conflicts. Unfortunately, such process takes time and occupies resource of the controller.

As depicted in Figure 6.3, on GolfEngine the rules produced by applications are not directly installed on the controller. Indeed, such rules are first checked by PCD, stored into the Database, and only then conflict-free rules are forwarded to the controller. By preventing to directly install potentially conflicting rules on the controller, PCD component of GolfEngine is once again more efficient than its competitors [190, 189].

However, it takes time to send the rules to the controller, hence the problem is to try to find the optimal number of rules sent to the controller to ensure the efficiency of GolfEngine. Table 6.1 shows the time spent for PCD sending conflict-free rules to the controller and installing on the switches in different scenarios (i.e., one rule, two rules, etc.). From the results in Table 6.1, we can observe that sending and installing batches of 20 conflict-free rules is a most time-efficient mechanism.

Table 6.1: Time spent to send rules to controller and install on switches.

Number of Rules	1	2	4	6	8	10
Time (ms)	0.5	0.79	1.53	2.53	3.2	3.6
Number of Rules	***20***	30	40	50	60	70
Time (ms)	***6.4***	12.25	20.55	36.25	56.04	61.4

6.5 Summary

In this chapter, we implemented and evaluated GolfEngine, our proposed architecture for SDN network management. To the best of our knowledge, this is the first work to support network manager to visually operate SDN tasks based on system level instead of module level. In GolfEngine architecture, we solved two critical issues by introducing novel components (PCD and NSC). One issue is how to efficiently detect and prevent potentially conflicting policies designed by users. Another issue is how to avoid the possible communication redundancy between the controller and OF switches. We implemented a prototype of our proposal on OpenDaylight controller, but it can be easily extended on other SDN controllers. As underlined by the results in our evaluation, GolfEngine platform ensures the robustness, the correctness and the efficiency of SDN network management systems.

Part III

Energy-Efficient Communication of IoT Devices

Chapter 7

Energy-Efficient Communication of IoT Devices

Internet of Things (IoT) is a novel paradigm that is rapidly gaining ground in the scenario of modern wireless telecommunications. In recent years, IoT is attracting significant attention from both academia and industry [196, 197]. IoT enables the communication between human users and smart devices, and also between smart devices themselves. IoT can be used in the fields of environmental monitoring [198], health-care [199], and smart cities [200]. Besides, it can be used to leverage sensor devices to gather valuable data [3] related to Human-Computer Interaction. IoT devices are widely used as effective transmission medium [201]. However, IoT devices considered in the aforementioned work are restricted in terms of battery energy, processing, memory, and transmission range.

Figure 7.1 shows a network structure of battery-powered HCI devices, in which, the battery-powered IoT devices connect with the outside world via one or several sinks/gateways. The sink is a base station, which is used to collect and process data in a centralized mode. In this chapter, we use "network", "sink" and "nodes" to represent the IoT network, the rendezvous point and battery-powered HCI devices, respectively.

In IoT, sensor are restricted by limited batteries and transmission range. Thus, if sensors cannot communicate with sink directly, they leverage other peers as forwarders to route traffic to sink. In such case, the nodes closer to the sink carry heavier traffic loads. As a result, inner layer nodes deplete their energy faster than other peers [202]. Consequently, a massive amount of energy (i.e., remaining energy in the devices far away from the sink) is wasted and the network lifetime ends prematurely. Wadaa et al. in [203] showed that by the time nodes closest to the sink deplete their energy, the farther nodes may still have 93% of their initial energy avail-

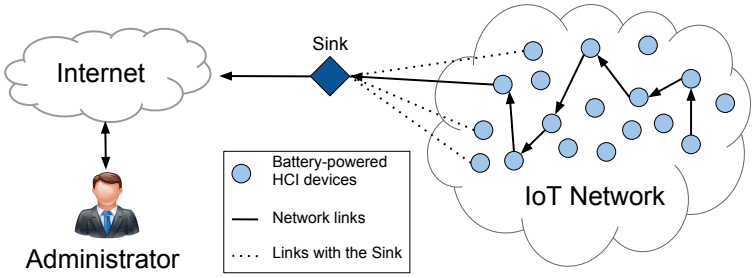


Figure 7.1: The structure of a network of battery-powered HCI devices.

able. This phenomenon is called *energy hole* problem [204]. In order to cope with *energy hole*, researchers have proposed several approaches based on duty cycle operation. Generally, nodes spend a large portion of power on monitoring surrounding environment instead of communicating with other nodes. This monitoring behavior is called *idle listening* [205]. In order to save energy by idle listening and balance the energy consumption, duty cycle ratio (i.e., periodically switching devices between sleeping mode and active mode) is being widely used in designing protocols for the IoT network of battery limited devices [206, 207, 208]. However, three issues exist in the energy conservation approaches applied with duty cycle ratios. First, the approaches assigning the same duty cycle ratio to all the nodes lead to a quick ending of network lifetime. According to the definition of *energy hole*, the inner nodes consumed more energy than the outer nodes and quickly ran out of their energy. Without the cooperation of inner nodes, the network cannot work properly and ends their lifetime quickly. Second, some protocols distribute different duty cycle ratios for different layers' nodes, however, they did not take the energy consumption during *network construction* phase. In other words, network lifetime is composed of network construction phase and data processing phase. Third, some of them focus on calculating energy consumption based on the concentric corona. However, compared with concentric corona structure, clustering structure is considered an effective approach to address the energy hole problem and prolong network lifetime [209, 210, 211]. One energy conservation approach is to prolong the network lifetime by adjusting the duty cycle ratios, unfortunately, it degrades network performance (i.e., increasing end-to-end delay). Therefore, we propose an energy conservative approach, EnergIoT, which could mitigate the *energy hole* problem and extend the network lifetime, without sacrificing network performance.

Chapter Organization. The rest of this chapter is organized as follows. We survey the state-of-the-art of the energy efficiency of battery-powered

HCI devices in Section 7.1. In Section 7.2, we first discuss the network structure of EnergIoT. Then, we describe the energy model considered in this chapter. In Section 7.3, we give a detailed description of our two objectives and we leverage mathematical method to get optimal values of different duty cycle ratios via our objectives. In Section 7.4, we report and discuss the result of the simulation. Finally, we conclude this chapter in Section 7.5.

7.1 Related Work

With the development of Internet technology and Wireless Sensor Networks (WSNs), a new paradigm for communication called Internet of Things is emerging in the era of ubiquity [212]. IoT network is full of smart "network enabled" objects, such as, smartphones, smart vehicles, buildings and other items embedded with electronics, software, sensors, actuators. Such smart objects have capabilities of sensing, acting, communicating, and processing advanced signal and information. IoT sensors play an important role in collecting, sending, and receiving a significant amount of data. Sensing and transmitting data are also key capabilities of WSNs. Therefore, most of the problems regarding the energy consumption in WSNs are inherited by IoT paradigms as well. In recent years, researchers have proposed numerous approaches aiming at optimizing energy consumption. The authors in [213] are the first to study how to avoid *energy hole* problem in WSNs. They proposed an energy consumption model governed by $E = d^{\alpha} + c$, where d is the transmission range, α is energy consumption coefficient and c is a positive constant parameter. They proved in theory that uneven energy depletion can be avoided when the width of all the coronas are same. C. Song et al. in [214] assigned different transmission ranges for different nodes to avoid *energy hole* problem. Based on the work [214], the authors in [215] found the optimal transmission path between each node and sink through ant colony algorithm. X. Wu et al. in [204] investigated the theoretical aspects of the non-uniform node distribution strategy. In particular, they regulated the number of nodes in each corona and derive the ratio q between the node densities in the adjacent $i+1$th and ith coronas by the strategy. However, they did not consider the energy consumption in *network construction* phase and idle listening consumption in *data processing* phase. The authors in [216, 217] proposed their own algorithm to find the most energy-efficient path to the sink for processing and storing data collected from sensors. Although all the aforementioned literature tried to solve the *energy hole* problem, some of them did not take the energy consumption caused by idle listening into consideration (e.g., [218]). In many scenarios, sensor nodes usually spend a considerable proportion of energy for monitoring the environment without communication, hence, the energy consumption of idle listening cannot be ignored. In order to reduce the energy consumption of

idle listening and balance energy consumption in different layers, the protocols of WSNs usually adopted duty cycle operation, which means nodes' working pattern alternates between sleeping and active mode. M. Medidi and Y. Zhou in [208] assigned different duty cycle ratios for nodes according to the distances from the sink. However, this model did not consider energy consumption caused in *network construction* phase, and did not rely on the clustering architecture that is efficient in energy conservation in a large scale network. Compared to the existing approaches, EnergIoT strategy proposed in this chapter not only considers energy consumption in *network construction* phase and *data processing* phase based on the hierarchical clustering structure, but it also assigns different duty cycle ratios for nodes to balance the energy consumption to cope with energy hole problem.

7.2 Model for EnergIoT

In order to present our EnergIoT, a hierarchical clustering approach based on different duty cycle ratios to improve the energy efficiency of a network of IoT devices, we introduce the network structure model for EnergIoT in Section 7.2.1. Based on the network structure, network lifetime is divided into *network construction* phase and *data processing* phase in EnergIoT (details in Section 7.2.2). In Section 7.2.3, we illustrate the energy model for calculating the energy consumption.

7.2.1 Network Structure Model

We assume that all the nodes are distributed in an area where the sink is located at the center of the network. All the sensor nodes are distributed within a distance R from the sink. The density of node distribution is ρ and the width of each layer is r that is the maximum transmission radius of sensors. We divide the network into m layers and set the ID of sink layer as 0, the ID of the outermost layer as m and so on. Since the maximum transmission range of a sensor node is r, the network is composed of a set of concentric coronas whose width is r. The data generated by nodes whose distance from the sink is larger than r is transmitted to the sink by multi-hop forwarding.

In EnergIoT, we manage and coordinate the nodes by leveraging hierarchical clustering architecture. Figure 7.2 shows the hierarchical clustering architecture. For ease of exposition, we show only a part of a whole network. A cluster consists of a cluster head (CH) and several cluster nodes (CNs). In particular, CNs are located in the same layer, exactly within the transmission range of the CH. CH is responsible for forwarding data gathered by its CNs to its inner layers' CH. Based on this structure, a node could play different roles in different clusters. For example, node a, located at layer 2,

acts as (i) a CH to receive data from its CNs located at layer 3, and (ii) a CN to transmit traffic to its CH located in layer 1. Recalling that a CH acts as a forwarder, we assume the ratio of the number of CH to the total CNs as μ. Moreover, each source node generates and sends λ bits of data per unit time.

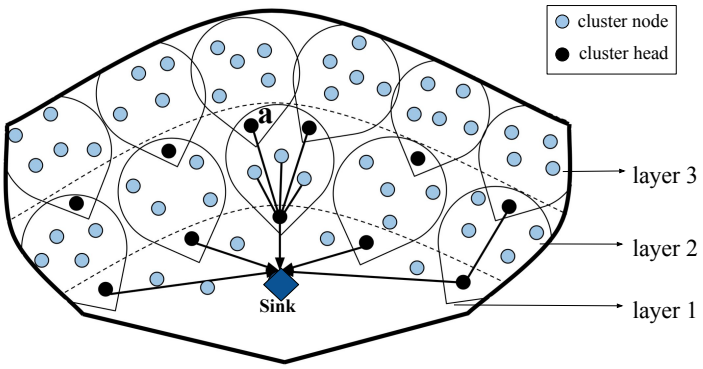


Figure 7.2: Hierarchical clustering architecture.

7.2.2 Network Workflow

Based on the above hierarchical clustering architecture, we give a description of the network workflow. The lifetime of the network consists of several rounds. When the number of dead sensor nodes (run out of energy) is larger than a threshold, the sink node starts the next round to "refresh" the *network construction* phase (see details in Section 7.2.2). Besides, each round includes two phases: *network construction* phase and *data processing* phase.

Network construction phase

During the *network construction* phase, EnergIoT uses multiple iterations to discover all the sensor nodes, and forms a hierarchical clustering architecture. The procedure of *network construction* phase is shown in Figure 7.3. In each iteration, nodes at layer i broadcast VH (Vie cluster Head) messages within range r to vie to be CH. A VH message includes the following information: remaining energy of the node and the identifier of the layer where the node is located (Step 1). Once a node located at layer $i+1$ receives VH messages from several CH candidates, it chooses the closest one and remains the more energy to be its CH. Then, the node sends a SH (Selecting cluster Head) message to the selected CH candidate (Step 2). The selected CH candidate at layer i replies AH (Ack cluster Head) messages to the nodes

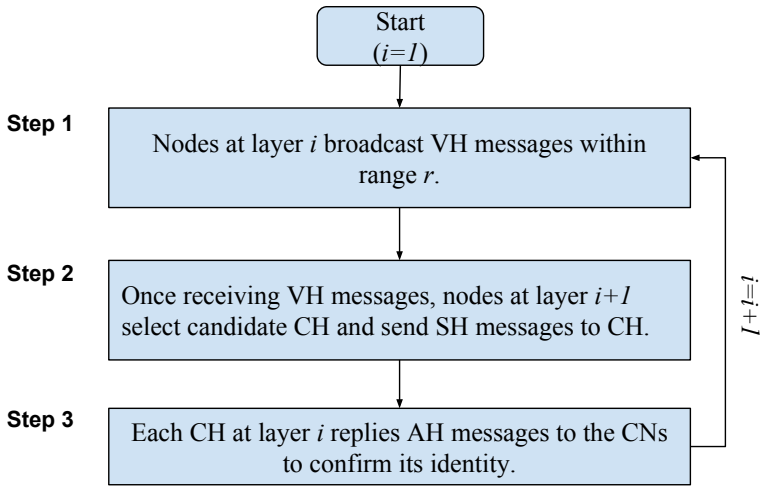


Figure 7.3: Procedure of network construction.

who are located at layer $i + 1$ and selects the candidate CH as their CH (Step 3).

Nodes at layer $i+1$ broadcast VH messages to nodes at layer $i+2$ to start a new iteration. Similarly, other CHs are found in different layers. Finally, in EnergIoT, after the *network construction* phase, we have the network where nodes are partitioned into layers according to their distances to the sink.

Data processing phase

During the *data processing* phase, each node in the network has two modes: *active* and *sleep*. In active mode, nodes consume energy to sense (i.e., idle listening), receive and transmit data. While in sleep mode, nodes are able to reduce energy consumption by turning off the sensing function. The duty cycle ratio γ, is defined as follows [205]:

$$\gamma = \frac{T_{active}}{T_{active} + T_{sleep}}$$

where T_{active} represents the duration of active mode and T_{sleep} is the duration of sleep mode. Note that, we can only adjust the duty cycle of a node by varying T_{sleep}, since T_{active} is fixed in order to guarantee receiving and transmitting network package [219].

7.2.3 Energy Consumption Model

In this section, we discuss the model for the possible energy consumption in EnergIoT. We assume that sensor nodes have the same initial energy, and the power of the sink is unlimited. First, we list the following types of energy consumption during *data processing* phase:

- E_{gl}: the average energy consumed for idle listening (active mode) in one unit time;
- E_{gr}: the average energy consumed for receiving data in one unit time;
- E_{gs}: the average energy consumed for transmitting data in one unit time.

In particular, E_{gl} is controlled by the duty cycle γ, the equation is listed in Eq. (7.1).

$$E_{gl} = \gamma e_{idle}. \tag{7.1}$$

Here, e_{idle} denotes a node's energy consumption for idle listening in per unit time [218]. Moreover, we set the value of e_{idle} to $0.88mJ/s$ for our EnergIoT. In order to calculate the energy consumption of sending and receiving packets (E_{gr} and E_{gs}), we list two basic formulas. The consumed energy in receiving l-bit data is shown in Eq. (7.2) and the consumed energy in sending l-bit data over a distance r is given by Eq. (7.3). Similar to the work [220], we assume a simple model where the radio dissipates $E_{elec} = 50nJ/bit$ to run the transmitter or receiver circuitry.

$$E_{gr} = lE_{elec}. \tag{7.2}$$

$$E_{gs} = l(E_{elec} + \varepsilon r^{\alpha}). \tag{7.3}$$

Second, we consider the following two types of energy consumption in *network construction* phase:

- E_{cl}: the energy consumed for idle listening in one unit time.
- E_{csr}: the energy consumed for receiving and transmitting messages (i.e., VH, SH and AH messages as described in Section 7.2.2) in one unit time.

In the following section, we illustrate concretely how to calculate the energy consumption in different layers and balance energy consumption among different layers.

7.3 Energy Consumption Optimization in EnergIoT

In this section, based on the network structure and energy model, we illustrate the main design of EnergIoT. In detail, we describe two objectives in

Section 7.3.1 and Section 7.3.2, respectively. According to the objectives, we take methods to find the optimal value of different duty cycle ratios for each layer in Section 7.3.3.

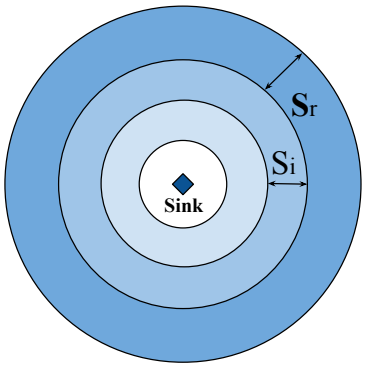
7.3.1 Balancing Energy Consumption

In order to accomplish this objective, we need to calculate the different type energy consumption of each layer's nodes in EnergIoT according to the order shown in Section 7.2.3.

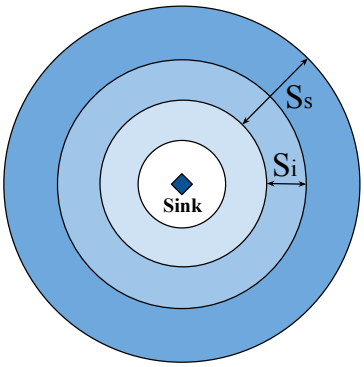
We use E_{gl}^{i} to represent the average idle listening energy consumption of CH in layer i in one unit time during *data processing* phase. The energy consumption for idle listening is also up to the ratio of CH. In EnergIoT, we assign different duty cycles for nodes at different layers. The duty cycle of a node at layer i is γ_i, and hence we have:

$$E_{gl}^{i} = \mu \gamma_i e_{idle}. \tag{7.4}$$

We denote E_{gr}^{i} as the average energy consumption of each node for receiving data at layer i in one unit time. In EnergIoT, a node at layer i receives data collected from all outer layers (i.e., from $i+1$th layer to mth layer). As shown in Figure 7.4(a), nodes in area S_i receive data generated from the nodes in area S_r. Note that, S_r includes one or more layers. Thus, according to Eq. (7.2), E_{gr}^{i} can be calculated as Eq. (7.5).



(a) S_i is in charge of receiving data from nodes in S_r.



(b) S_i is in charge of sending data generated by nodes in S_s.

Figure 7.4: Diagram to show areas S_i, S_r, and S_s.

$$\begin{aligned} E_{gr}^{i} &= lE_{elec} = \frac{S_r \rho \lambda}{S_i \rho \mu} E_{elec} = \frac{\pi[(mr)^2 - (ir)^2]\rho\lambda}{\pi[(ir)^2 - ((i-1)r)^2]\rho\mu} E_{elec} \\ &= \frac{\lambda(m^2 - i^2)}{\mu(2i-1)} E_{elec} \quad (1 \le i \le m). \end{aligned} \tag{7.5}$$

Here, $S_r\rho\lambda$ represents the data generated by the nodes in area S_r in one unit time. $S_i\rho\mu$ is the number of CH responsible for receiving data in area S_i. Hence, $\frac{S_r\rho\lambda}{S_i\rho\mu}$ is the average data received by each CH in area S_i in one unit time.

We use E^i_{gs} to express the average energy consumption of a CH node for transmitting data at layer i in one unit time. In EnergIoT, CHs at layer i not only transmit data generated by themselves, but also forward the data generated by nodes from all outer layers. As shown in Figure 7.4(b), nodes in area S_i transmit data generated by nodes from S_s that includes area S_i. Therefore, according to Eq. (7.3), E^i_{gs} can be calculated as follows:

$$\begin{aligned} E^i_{gs} &= l(E_{elec} + \varepsilon r^\alpha) = \frac{S_s\rho\lambda}{S_i\rho\mu}(E_{elec} + \varepsilon r^\alpha) \\ &= \frac{\pi[(mr)^2 - ((i-1)r)^2]\rho\lambda}{\pi[(ir)^2 - ((i-1)r)^2]\rho\mu}(E_{elec} + \varepsilon r^\alpha) \\ &= \frac{\lambda[m^2 - (i-1)^2]}{\mu(2i-1)}(E_{elec} + \varepsilon r^\alpha) \quad (1 \le i \le m). \end{aligned} \tag{7.6}$$

Here, $S_s\rho\lambda$ represents the data generated by the nodes in area S_s in one unit time. $S_i\rho\mu$ denotes the number of nodes responsible for transmitting data in area S_i. Hence, $\frac{S_s\rho\lambda}{S_i\rho\mu}$ is the average data sent by each CH in area S_i in one unit time.

We denote E^i_{cl} as the average idle listening energy consumption of CH node at layer i in one unit time during *network construction* phase. As described in Section 7.2.2, during the cluster construction, nodes send VH messages to outer layers. Once these nodes send AH message to confirm that they are CHs, they enter into sleep mode. Therefore, we have Eq. (7.7):

$$E^i_{cl} = \frac{i}{m-1}e_{idle} \qquad 1 \le i \le m. \tag{7.7}$$

There are $m-1$ rounds to choose the CHs during network construction phase. Nodes in layer i need to wait for i rounds until they send AH message and enter into sleep mode.

We use E_i to denote the average energy consumption per unit time of a CH node in layer i in EnergIoT, and here we describe it in Eq. (7.8).

$$\begin{aligned} E_i &= E^i_{gl} + E^i_{gr} + E^i_{gs} + E^i_{cl} + E^i_{csr} \\ &= \mu\gamma_i e_{idle} + \frac{\lambda(m^2 - i^2)}{\mu(2i-1)}E_{elec} + \frac{\lambda[m^2 - (i-1)^2]}{\mu(2i-1)}(E_{elec} + \varepsilon r^\alpha) \\ &+ \frac{i}{m-1}e_{idle} + E^i_{csr} \qquad (1 \le i \le m). \end{aligned} \tag{7.8}$$

Because of the *energy hole* phenomenon, we try to balance the energy consumption among different layers' CH nodes using the Eq. (7.9) to prolong network lifetime.

$$E_i \approx E_{i+1}. \tag{7.9}$$

It is deserved to notice that, during the *network construction* phase, the number of sending and receiving messages by nodes at different layers is same. The expression in equation is $E^i_{csr} = E^{i+1}_{csr}$, hence the expansion of Eq. (7.9) is as follows.

$$E^i_{gl} + E^i_{gr} + E^i_{gs} + E^i_{cl} \approx E^{i+1}_{gl} + E^{i+1}_{gr} + E^{i+1}_{gs} + E^{i+1}_{cl}. \tag{7.10}$$

7.3.2 Maintaining Network Performance

In order to guarantee the network performance after introducing EnergIoT, we should keep the same end-to-end delay as that of uniform duty cycle approach. In both of EnergIoT and uniform duty cycle approach, the end-to-end delay consists of the setup latency and the transmission delay. We try to analyze the delay of a packet sent from the outermost layer, because the end-to-end delay of this situation is highest. As we mentioned in Section 7.2.1, the duty cycle ratios of nodes are controlled by varying the sleep interval T_{sleep}. The active period T_{active} is fixed for each node in order to ensure receiving data packets. Therefore, a node with value of duty cycle γ_i generates a setup latency T_{st} [219]. $T_{st} = T_{sleep}/2K = T_{active}(1-\gamma_i)/2K\gamma_i$, where K is the average number of forwarders each sender have. K is set to 1 according to our structure model in Section 7.2.1.

We use D_e, D_u to represent the end-to-end delay in EnergIoT and uniform duty cycle ratio approach, respectively. T^i_{st} is the setup latency for layer i, while T_{tran} is the transmission delay of sending a packet from layer i to $i+1$. Therefore, D_e and D_u can be represented as follows:

$$D_e = \sum_{i=1}^{m} T^i_{st} + mT_{tran} = T_{active} \sum_{i=1}^{m} \frac{1-\gamma_i}{2\gamma_i} + mT_{tran}, \tag{7.11}$$

$$D_u = \sum_{i=1}^{m} T^i_{st} + mT_{tran} = mT_{active} \frac{1-\gamma}{2\gamma} + mT_{tran}. \tag{7.12}$$

In order to maintain network performance of end-to-end delay in EnergIoT with uniform duty cycle ratios approach, we must make sure that the end-to-end delay in EnergIoT is less than or equal to that in uniform duty cycle ratio approach. Thus, we have Eq. (7.13):

$$D_e \leqslant D_u \Rightarrow m\frac{1-\gamma}{\gamma} \leqslant \sum_{i=1}^{m} \frac{1-\gamma_i}{\gamma_i}. \tag{7.13}$$

We can notice density node distribution ρ is canceled out from Eq. (7.10) and (7.13), hence the optimization of energy consumption has nothing to do with ρ.

7.3.3 Value of Different Duty Cycle Ratios

In this section, we formulate optimization problem to find the optimal different duty cycle ratios to achieve the objectives mentioned in Section 7.3.1 and 7.3.2. We have two conditions provided in Eq. (7.10) and (7.13) for γ_i. As our prime objective is to improve the energy efficiency, we formulate our objective function based on energy criteria and constraint function based on the end-to-end delay. Therefore, we try to calculate the minimal value of our objective function (Eq. (7.14)) such that the optimal value of γ_i can satisfy constraint function (Eq. (7.15)). The constrained optimization problem is expressed as

$$\min_{\gamma_1,\gamma_2,...,\gamma_m} \quad f(\gamma_1,\gamma_2,...,\gamma_m) = \sum_{i=1}^{m}(E_i - E_{i+1})^2, \tag{7.14}$$

such that

$$m\frac{1-\gamma}{\gamma} \leqslant \sum_{i=1}^{m}\frac{1-\gamma_i}{\gamma_i}. \tag{7.15}$$

Relying on Lagrange Multiplier [221, 222], we find the optimal values for variables $(\gamma_1,\gamma_2,...,\gamma_m)$ and satisfy our constraint at the same time.

We observe that the closer the nodes to the sink, the smaller the the duty cycle ratio. Because the nodes closer to the sink spend more energy to forward and receive the data, they need more time for sleep mode. According to the definition of γ, under the condition of fixed T_{active}, the nodes closer to sink have a smaller value for duty cycle ratios. EnergIoT effectively mitigates the problem of unbalanced energy consumption, which is also demonstrated in our experiments (discussed in Section 7.4).

7.4 Simulation Analysis

In this section, we carry out extensive simulations to validate the effectiveness of EnergIoT. In particular, we compare EnergIoT with the approach without duty cycle mechanism (we refer this approach as *full-active* approach in the following), as well as the approach with uniform duty cycle ratios (we refer this approach as *uniform* approach in the following). The fixed duty cycle of uniform approach is 2% in the simulation. Our simulation is performed by Castalia Emulator [223] on OMNeT++ platform [224]. The metrics used in the simulation are shown in Table 7.1.

In the following, we first compare the lifetime of the network in three strategies. Then we show the percentage of remaining energy in different layers of the network for these three approaches. At last, we give a description of the end-to-end delay in different layers of the network. Moreover, we report the average results obtained from 50 simulation experiments and the figures are with 95% confidence interval error bars.

Table 7.1: Parameters in EnergIoT's simulation scenario.

Model	**Parameter**	**Value**
Energy Model	E_{elec}	50nJ/bit
	α	2
	ε	$100pJ/bit/m^2$
	e_{idle}	0.88mJ/s
	γ	2%
	Initial energy of each node	$10.8 * 10^3 J$
Network Model	r	20m
	R	200m
	m	10
	μ	20%
	ρ	$60/m^2$
	λ	400bps

7.4.1 Lifetime Analysis

The average lifetime of the network in EnergIoT, *uniform* approach and *full-active* approach is around 361.42 hours, 272.87 hours and 53.3 hours, respectively. The result shows that the approaches deployed with duty cycle could significantly prolong the lifetime of the network. The reason is that, in EnergIoT and *uniform* approach, nodes turn off the sensing function and enter into sleeping mode periodically, which contributes to reducing the energy consumption. However, in *full-active* approach, the nodes closest to sink deplete their energy so quickly that the network ends the life soon. Although the *uniform* approach achieves the aim of energy conservation, it sacrifices the end-to-end delay (details in Section 7.4.3). Moreover, the network lifetime with EnergIoT strategy extends 32% compared with that of *uniform* approach. Since the energy consumption of nodes in different layers is balanced by introducing different duty cycle ratios, the network will not end prematurely because of *energy hole* phenomenon.

7.4.2 Energy Remaining in Different Layers

As a result of our experiment, we highlight in Figure 7.5 how EnergIoT outperforms the other two approaches in terms of energy efficiency. Indeed,

the energy efficiency of a network is given by the percentage of remaining energy in its nodes. As shown in Figure 7.5, the percentage of remaining

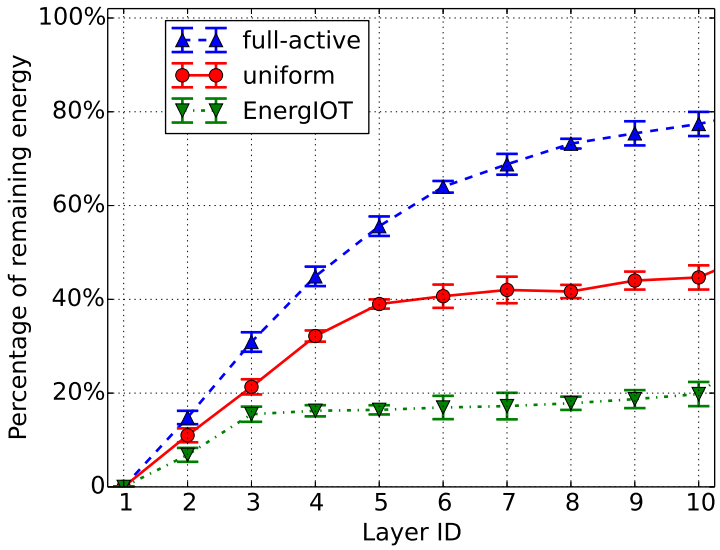


Figure 7.5: Comparison of percentage of remaining energy.

energy increases with the increasing ID in these three approaches. In both of *uniform* approach and *full-active* approach, the percentage of remaining energy is almost 80% and 45% respectively for most of the layers while the first layer nodes run out of their energy. However, when the layer ID increases, in EnergIoT, the percentage of remaining energy fluctuates around 20% for almost all layers except the first and second layers. We can conclude that the energy consumption is balanced well through different duty cycle ratios. The nodes located at inner layers (closer to the sink) are responsible for receiving and transmitting more traffic loads. If we do not take measure to balance the energy consumption, the inner nodes will deplete energy quickly, which leads to the outer layers' nodes remaining amount of energy (i.e., 45%, 80%). It is a waste for the outer layers nodes to remain such amount of energy.

In EnergIoT, we deploy different duty cycle ratios for nodes in different layers. In particular, the nodes in the inner layer have smaller duty cycle ratios, which represents that the nodes can sleep more time when it is not in active mode. As we discussed in Eq. (7.10) and (7.13), the energy consumption in different layers is approximately same by tuning the duty cycle ratios. Moreover, the remaining energy of nodes with *full-active* approach is still more or less 80%. The reason for this phenomenon is that the *full-active* approach does not consider turning off the sensing model of nodes. Hence, the inner layers' nodes are easily and quickly run out of their energy.

7.4.3 End-to-End Delay Analysis

End-to-end delay consists of time to transmit packets from a source to the destination and the latency of setting up duty cycle ratios for different layers' nodes. In our simulation, we report the average end-to-end delay, which represents delay performance. Figure 7.6 shows the end-to-end comparison among EnergIoT, *uniform* approach, *full-active* approach. As shown in

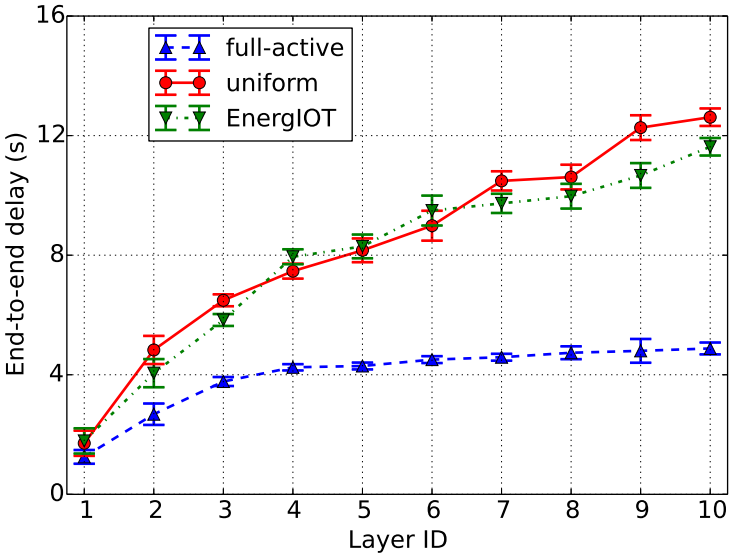


Figure 7.6: Comparison of end-to-end delay.

Figure 7.6, the end-to-end delay of the network with *full-active* approach is less than the network with duty cycle-based approaches. One reason is that, with *full-active* approach, nodes spend all time in the idle listening mode instead of sleeping mode. Therefore, nodes with *full-active* approach can response the traffic once such traffic arrives without wasting much time. The other reason is that, the latency for setting up duty cycle is 0, since γ is 1 (Section 7.3.2). However, in duty cycle-based approaches, the nodes might be in the sleep mode when the traffic arrives, which means nodes cannot respond immediately to any traffic.

The end-to-end delay in both EnergIoT and *uniform* approach is more or less similar as shown in Figure 7.6. We can deduce that EnergIoT achieves the target of extending network lifetime of IoT devices without sacrificing network performance compared to *uniform* approach. The simulation result confirms that EnergIoT efficiently improves network lifetime without prolonging end-to-end delay. In addition, Figure 7.6 shows that the nodes located at outer layers have longer end-to-end delay compared with nodes in the inner layers (i.e., the end-to-end delay increases with the increasing ID). EnergIoT is a differential duty cycle approach based on hierarchical cluster-

ing architecture. The network is composed of multiple clusters consisting of CH and CNs. The CNs send the packets to their CH directly. Therefore, nodes at outer layers need more forwarders to transmit packets. As a result, the end-to-end delay of nodes at outer layers is larger.

7.5 Summary

In this chapter, we proposed EnergIoT, a hierarchical clustering approach, based on different duty cycle ratios to maximize the lifetime of IoT network of battery-powered devices, under the condition of maintaining the network performance. In particular, we assigned the different duty cycle ratios to nodes according to their different distance from the sink. First, the energy consumption of the battery-powered devices are balanced especially for the innermost layer and outermost layer, which avoids the *energy hole* problem. Second, EnergIoT keeps the network performance (i.e., delay) the same as uniform duty cycle approach. Moreover, we modeled the energy consumption considering both *data processing* phase and *network construction* phase based on a hierarchical clustering network structure. We assessed EnergIoT through extensive simulation-based analyses on the OMNet++ platform. The results showed that EnergIoT is feasible and efficient. EnergIoT is able to improve the lifetime of IoT network by 32%, when compared to the approach applied with uniform duty cycle, without degrading the network performance.

Chapter 8

Conclusions

Considerable researches have been carried out and major advances have been made in the area of Human-Computer Interaction. Information security and authentication are becoming increasingly important and more complex as business is conducted electronically. However, the state-of-the-art of security-related development still needs to be improved in the aspect of HCI. The objective of this dissertation is to promote and enable security awareness of end-users in their interaction with computer systems.

In this dissertation, we presented our contribution in security aspects of HCI. In what follows, we summarize our contribution (Section 8.1), and discuss future research directions (Section 8.2).

8.1 Summary of Contribution

We described the contributions of this dissertation in three parts: (1) an illustration of HCI-based authentication and a keystroke dynamics based identification of users; (2) a study of HCI impact on network traffic and a framework, GolfEngine, for managing such network traffic to ensure network without malicious attack of human behavior; and (3) a proposal, EnergIoT, for avoiding energy exhaustion attacks and further extending the lifetime of networks of battery-powered IoT smart devices.

8.1.1 Security Issues in User Interaction with Devices

Part I presented our work about the security challenge of authentication users on smart devices. In particular, we designed a tool based on keystroke dynamics to detect users' deceptive information.

- *HCI-based Authentication for Smartphones:* Chapter 2 surveyed the state-of-the-art of authentication methods on smartphones that are based on HCI. First, we categorized authentication methods according to the nature of biometrics features, which are measured by smartphone sensors, into behavioral and physiological. Physiological biometrics are related to users' characteristics, while behavioral biometrics are related to the way a user interacts with her smartphone. Second, we categorized the behavioral authentication methods into two types according to the number of time required to collect data from users and authenticating. Finally, we discussed the future research directions of biometrics authentication methods on smartphones.

- *HCI-based Security Challenges in Brain-Computer Interfaces:* BCI is deployed to translate the brain activity to signals that can be used to control computers or communication devices. In Chapter 3, we surveyed main common brain-computer interfaces (BCI) applications and categorized them into four different application scenarios: (1) neuromedical applications; (2) user authentication; (3) gaming and entertainment; and (4) smartphone-based applications. For each scenario, we provide detailed description of the state-of-the-art technologies, potential attacks that might threat each scenario, and envisaged countermeasures.

- *Detecting Deceitful Users via Keystroke Dynamics:* Interacting with a computer in a natural way is a key issue for decades. The most commonly known interfaces to control computer and other devices are keyboards. In Chapter 4, we presented a new method based on keystroke dynamics to distinguish between fake and truthful personal information written via a computer keyboard. Our authentication method reached an overall accuracy of 76% in the classification of a single answer as truthful or deceptive.

8.1.2 HCI Impact on Network Traffic

Part II illustrated the influence of the users' behavior on network traffic and proposed an architecture to manage such network traffic efficiently and securely.

- *Analysis of Network Traffic on Mobile Devices:* We reviewed the work in Chapter 5 that contributes to the state of the art of network traffic analysis targeting mobile devices. In particular, we present a systematic classification of the work in the literature according to three criteria: (i) the goal of the analysis; (ii) the point where the network traffic is captured; and (iii) the targeted mobile platforms. In particular, we illustrate detailed the goal of the analysis related to Human-Computer

Interaction. We believe our survey will be an important reference work for researchers and practitioners in this research field.

- *Network Traffic Management via SDN:* In Chapter 6, we proposed an innovative framework named GolfEngine, based on SDN, to simplify the management of network traffic and deployment of security applications to manage the user behavior. GolfEngine provided better visibility to dominate tasks for performing network anomalies on demand. In addition, GolfEngine provided function of checking policy conflicts when users design their own security applications. Furthermore, our GolfEngine provided the mechanism to check possible redundancies on data storage of network traffic generated by human behavior, which improved the network security application efficiency. Moreover, we evaluated the performance and execution efficiency of GolfEngine through a security use case (i.e., Firewall) implementation. The results of our simulation underlined that GolfEngine contributes significantly in improving the efficiency and security of management HCI impact on network.

8.1.3 Energy-Efficient Communication of IoT Devices

In part III, in Chapter 7, we proposed EnergIoT, a hierarchical clustering approach based on duty cycle ratio to maximize network lifetime of battery-powered HCI based devices. In particular, EnergIoT is designed to automatically assign different duty cycle ratios to devices according to their distance from the sink, since different duty cycle ratios balance the energy consumption among devices at different layers. Furthermore, we calculated the energy consumption of HCI based devices, considering both *network construction* phase and *data processing* phase. We evaluated EnergIoT through extensive simulation analyses on the OMNet++ platform. The result showed that EnergIoT was not only feasible but also efficient. Moreover, EnergIoT improved the network lifetime of IoT smart devices by 32%, compared to the uniform duty cycle approach, without sacrificing the network performance (i.e., end-to-end delay).

8.2 Future Work

In this section, we discuss possible future developments that naturally follow from the research contribution presented in this dissertation.

8.2.1 Security Issues in User Interaction with Devices

- *HCI-based Security Challenges in Brain-Computer Interfaces:* In Chapter 3, we pointed out the current attacks and corresponding measurements for each scenario's BCI applications. In order to strengthen

the security aspects of BCI, we intend to focus on countermeasures to prevent the use of BCI capabilities to perpetrate user privacy attacks. As shown in Figure 8.1, our idea is to add a process named "BCI Protector" between the signal acquisition and signal digitization. The core design of BCI protector is to make a synthetic signal by combining the raw brain signal with a regular wave (such as sine or cosine wave). In this way, the attackers cannot recognize the raw signal from synthetic signal.

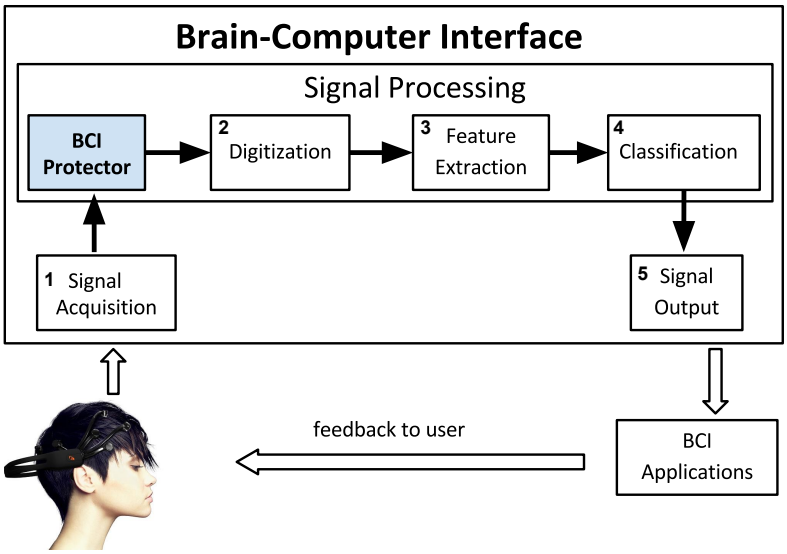


Figure 8.1: The mechanism to prevent attacks for BCI.

- *Detecting Deceitful Users via Keystroke Dynamics:* In Chapter 4 proved that keystroke analysis had a great potential in detecting the veracity of self-declared personal information. In order to enforce the authentication when interacting with the keyboard, our future work will try to combine the brain activities information together with the keystroke dynamics as user behavior data for authentication. Moreover, we will investigate the feasibility to carry out false information to rely on brain activity data provided by the devices described in Chapter 3.

8.2.2 HCI Impact on Network traffic

The management architecture of network traffic generated by users' interaction with smart devices was introduced in Chapter 6. Future work is focused on two main directions. On one hand, analyzing the data provided by the database of "User Information", we aim at predicting users' behavior, being able to design in advance security applications that prevent possible

errors and attacks. On the other hand, in the future, we intend to face the challenges on multi-domain network traffic.

8.2.3 Energy-Efficient Communication of IoT Devices

As a future work for our proposal in Chapter 7, we will further investigate the impact of each component on the overall performance in terms of energy consumption and communication delay reduction. In particular, we intend to run ad-hoc simulations to adjust the percentage of CHs in a network. Moreover, we would like to evaluate the efficiency of our proposal in a real-world IoT testbed.

Bibliography

[1] Gaurav Sinha, Rahul Shahi, and Mani Shankar. Human computer interaction. In *Proceedings of the 3rd International Conference on Emerging Trends in Engineering and Technology (ICETET)*, pages 1–4. IEEE, 2010.

[2] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. *Research methods in human-computer interaction*. Morgan Kaufmann, 2017.

[3] QianQian Li, Ding Ding, and Mauro Conti. Brain-Computer Interface applications: Security and privacy challenges. In *Proceddings of Communications and Network Security (CNS)*, pages 663–666. IEEE, 2015.

[4] Riccardo Spolaor, QianQian Li, Merylin Monaro, Mauro Conti, et al. Biometric authentication methods on smartphones: A survey. *PsychNology Journal*, 14(2-3):87–98, 2016.

[5] Merylin Monaro, Riccardo Spolaor, QianQian Li, Mauro Conti, et al. Type me the truth!: Detecting deceitful users via keystroke dynamics. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, page 60. ACM, 2017.

[6] QianQian Li, Reza Mohammadi, Mauro Conti, Chuanhuang Li, and Xiaolin Li. Golfengine: Network management system for software defined networking. In *Proceedings of the 13th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2017.

[7] QianQian Li, Sarada Prasad Gochhayat, Mauro Conti, and FangAi Liu. Energiot: A solution to improve network lifetime of iot devices. *Pervasive and Mobile Computing*, 2017.

[8] Mauro Conti, QianQian Li, Alberto Maragno, and Riccardo Spolaor. The dark side(-channel) of mobile devices: A survey on network traffic analysis. *Communications Surveys and Tutorials*, 2017.

[9] Alexander De Luca, Marian Harbach, Emanuel von Zezschwitz, et al. Now you see me, now you don't: protecting smartphone authentication from shoulder surfers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2937–2946. ACM, 2014.

[10] Adam J Aviv, Katherine L Gibson, Evan Mossop, Matt Blaze, and Jonathan M Smith. Smudge attacks on smartphone touch screens. *Woot*, 10:1–7, 2010.

[11] Hugo Gamboa and Ana LN Fred. An identity authentication system based on human computer interaction behaviour. In *PRIS*, pages 46–55, 2003.

[12] Anirudh Vallabhaneni, Tao Wang, and Bin He. Braincomputer interface. *Neural engineering*, pages 85–121, 2005.

[13] Pin Shen Teh, Andrew Beng Jin Teoh, and Shigang Yue. A survey of keystroke dynamics biometrics. *The Scientific World Journal*, 2013.

[14] G Mark Grimes, Jeffrey L Jenkins, and Joseph S Valacich. Assessing credibility by monitoring changes in typing behavior: The keystroke dynamics deception detection model. In *HICSS-46 Symposium on Credibility Assessment and Information Quality in Government and Business*, 2013.

[15] Ritwik Banerjee, Song Feng, Jun Seok Kang, and Yejin Choi. Keystroke patterns as prosody in digital writings: A case study with deceptive reviews and essays. In *International Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1469–1473, 2014.

[16] Jan De Houwer, Sarah Teige-Mocigemba, Adriaan Spruyt, and Agnes Moors. Implicit measures: A normative analysis and review. *Psychological bulletin*, 135(3):347, 2009.

[17] Hyojoon Kim, Theophilus Benson, Aditya Akella, and Nick Feamster. The evolution of network configuration: a tale of two campuses. In *Proceedings of ACM SIGCOMM conference on Internet measurement conference*, pages 499–514. ACM, 2011.

[18] Nancy Gibbs. Your life is fully mobile. `http://techland.time.com/2012/08/16/your-life-is-fully-mobile/`.

[19] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *IEEE Communications magazine*, 48(9), 2010.

[20] Samsung GRO proposal. `http://www.sait.samsung.co.kr/`.

[21] Identify real users continuously and without friction. `http://www.biocatch.com/behavioral-authentication`.

[22] Abdenour Hadid, JY Heikkila, Olli Silvén, and M Pietikainen. Face and eye detection for person authentication in mobile phones. In *Proceedings of 1st ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pages 101–108. IEEE, 2007.

[23] Kiran B Raja, Ramachandra Raghavendra, Martin Stokkenes, and Christoph Busch. Smartphone authentication system using periocular biometrics. In *International Conference on Biometrics Special Interest Group (BIOSIG)*, pages 1–8. IEEE, 2014.

[24] PN Ali Fahmi, Elyor Kodirov, Deok-Jai Choi, Guee-Sang Lee, A Mohd Fikri Azli, and Shohel Sayeed. Implicit authentication based on ear shape biometrics using smartphone camera during a call. In *International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2272–2276. IEEE, 2012.

[25] Kasper Bonne Rasmussen, Marc Roeschlin, Ivan Martinovic, and Gene Tsudik. Authentication using pulse-response biometrics. In *The Network and Distributed System Security Symposium (NDSS)*, 2014.

[26] Dong-Ju Kim, Kwang-Woo Chung, and Kwang-Seok Hong. Person authentication using face, teeth and voice modalities for mobile device security. *IEEE Transactions on Consumer Electronics*, 56(4), 2010.

[27] Arman Boehm, Dongqu Chen, Mario Frank, Ling Huang, et al. Safe: Secure authentication with face and eyes. In *International Conference on Privacy and Security in Mobile Systems (PRISMS)*, pages 1–8. IEEE, 2013.

[28] Cristiano Giuffrida, Kamil Majdanik, Mauro Conti, and Herbert Bos. I sensed it was you: authenticating mobile users with sensor-enhanced keystroke dynamics. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 92–111. Springer, 2014.

[29] Valeriu-Daniel Stanciu, Riccardo Spolaor, Mauro Conti, and Cristiano Giuffrida. On the effectiveness of sensor-enhanced keystroke dynamics against statistical attacks. In *Proceedings of the Sixth ACM Conference*

on Data and Application Security and Privacy, pages 105–112. ACM, 2016.

[30] Nan Zheng, Kun Bai, Hai Huang, and Haining Wang. You are how you touch: User verification on smartphones via tapping behaviors. In *Proceedings of the 22nd IEEE International Conference on Network Protocols (ICNP)*, pages 221–232. IEEE, 2014.

[31] Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 987–996. ACM, 2012.

[32] Hataichanok Saevanee and Pattarasinee Bhatarakosol. User authentication using combination of behavioral biometrics over the touchpad acting like touch screen of mobile device. In *Proceedings of IEEE International Conference on Computer and Electrical Engineering (ICCEE)*, pages 82–86. IEEE, 2008.

[33] Mauro Conti, Irina Zachia-Zlatea, and Bruno Crispo. Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 249–259. ACM, 2011.

[34] Jani Mantyjarvi, Mikko Lindholm, Elena Vildjiounaite, et al. Identifying users of portable devices from gait pattern with accelerometers. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, volume 2, page 973. IEEE, 2005.

[35] Mohammad Omar Derawi, Claudia Nickel, Patrick Bours, and Christoph Busch. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *Proceedings of Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, pages 306–311. IEEE, 2010.

[36] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE transactions on information forensics and security*, 8(1):136–148, 2013.

[37] Hugo Gascon, Sebastian Uellenbeck, Christopher Wolf, and Konrad Rieck. Continuous authentication on mobile devices by analysis of typing motion behavior. In *Sicherheit*, pages 1–12, 2014.

[38] Elaine Shi, Yuan Niu, Markus Jakobsson, and Richard Chow. Implicit authentication through learning user behavior. In *ISC*, volume 6531, pages 99–113. Springer, 2010.

[39] Mauro Conti, Bruno Crispo, Earlence Fernandes, and Yury Zhauniarovich. Crêpe: A system for enforcing fine-grained context-related policies on android. *IEEE Transactions on Information Forensics and Security*, 7(5):1426–1438, 2012.

[40] Mauro Conti, Luigi V Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. Can't you hear me knocking: Identification of user actions on android apps via traffic analysis. In *DASP*, pages 297–304, 2015.

[41] CAPTCHA. `https://www.capy.me/products/puzzle_captcha/`.

[42] Mauro Conti, Claudio Guarisco, and Riccardo Spolaor. Captchastar! a novel captcha based on interactive shape discovery. In *International Conference on Applied Cryptography and Network Security*, pages 611–628. Springer, 2016.

[43] Jagmohan Chauhan, Hassan Jameel Asghar, Anirban Mahanti, and Mohamed Ali Kaafar. Gesture-based continuous authentication for wearable devices: The smart glasses use case. In *International Conference on Applied Cryptography and Network Security*, pages 648–665. Springer, 2016.

[44] Vincent F Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *Proceedings of IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 439–454. IEEE, 2016.

[45] Mauro Conti, Luigi Vincenzo Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. Analyzing android encrypted network traffic to identify user actions. *IEEE Transactions on Information Forensics and Security*, 11(1):114–125, 2016.

[46] Mauro Conti, Michele Nati, Enrico Rotundo, and Riccardo Spolaor. Mind the plug! laptop-user recognition through power consumption. In *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security*, pages 37–44. ACM, 2016.

[47] Michal Teplan. Fundamentals of eeg measurement. *Measurement science review*, 2(2):1–11, 2002.

[48] Jonathan R Wolpaw, Niels Birbaumer, William J Heetderks, et al. Brain-computer interface technology: a review of the first international meeting. *IEEE transactions on rehabilitation engineering*, 8(2):164–173, 2000.

[49] Juri Kropotov. *Quantitative EEG, event-related potentials and neurotherapy*. Academic Press, 2010.

[50] Biosemi. `http://www.biosemi.com`.

[51] Emotiv epoc. `https://emotiv.com`.

[52] Neurosky. `http://neurosky.com`.

[53] Tamara Denning, Yoky Matsuoka, and Tadayoshi Kohno. Neurosecurity: security and privacy for neural devices. *Neurosurgical Focus*, 27(1):E7, 2009.

[54] Andrew B Schwartz, X Tracy Cui, Douglas J Weber, and Daniel W Moran. Brain-controlled interfaces: movement restoration with neural prosthetics. *Neuron*, 52(1):205–220, 2006.

[55] John Chuang, Hamilton Nguyen, Charles Wang, and Benjamin Johnson. I think, therefore i am: Usability and security of authentication using brainwaves. In *Financial Cryptography and Data Security*, pages 1–16. 2013.

[56] Abhejit Rajagopal, Anthony C Nguyen, and Dennis M Briggs. Neuropass: A secure neural password based on eeg. *Biomedical Engineering*, 2013.

[57] Christian Mühl, Hayrettin Gürkök, Danny Plass-Oude Bos, et al. Bacteria hunt: A multimodal, multiparadigm bci game. *University of Genua*, 2010.

[58] Marco Congedo, Matthieu Goyat, Nicolas Tarrin, Gelu Ionescu, et al. Brain invaders: a prototype of an open-source p300-based video game working with the openvibe platform. In *5th International BCI*, pages 280–283, 2011.

[59] Andrea Finke, Alexander Lenhardt, and Helge Ritter. The mindgame: a p300-based brain-computer interface game. *Neural Networks*, 22(9):1329–1333, 2009.

[60] Alexander Kaplan, S Shishkin, I Ganin, I Basyul, and A Zhigalov. Adapting the p300-based brain-computer interface for gaming: a review. *CI and AI in Games*, 5(2):141–149, 2013.

[61] Erik Andreas Larsen. Classification of eeg signals in a brain-computer interface system. Norwegian University, 2011.

[62] Andrew Campbell, Tanzeem Choudhury, Shaohan Hu, Hong Lu, et al. Neurophone: brain-mobile phone interface using a wireless eeg headset. In *Proceedings of the second ACM SIGCOMM workshop*, pages 3–8, 2010.

[63] Jung-Yoon Kim and Won-Hyung Lee. Eeg signal feature analysis of smartphone game user. *ASTL*, 39:14–19, 2013.

[64] Keita Honda and Suguru N Kudoh. Air brain: the easy telemetric system with smartphone for eeg signal and human behavior. In *Proceedings of the 8th BodyNets*, pages 343–346, 2013.

[65] Kendall Lee, David Roberts, and Alexander Hartov. Deep brain stimulator, October 5 2005. US Patent App. 11/244,134.

[66] Robert J Coffey. Deep brain stimulation for chronic pain: results of two multicenter trials and a structured review. *Pain Medicine*, 2(3):183–192, 2001.

[67] Günther Deuschl, Carmen Schade-Brittinger, Paul Krack, Jens Volkmann, et al. A randomized trial of deep-brain stimulation for parkinson's disease. *New England Journal of Medicine*, 355(9):896–908, 2006.

[68] David M Santucci, Jerald D Kralik, Mikhail A Lebedev, and Miguel AL Nicolelis. Frontal and parietal cortical ensembles predict single-trial muscle activity during reaching movements in primates. *European Journal of Neuroscience*, 22(6):1529–1540, 2005.

[69] Benjamin Johnson, Thomas Maillart, and John Chuang. My thoughts are not your thoughts. In *Proceedings of UbiComp: Adjunct Publication*, pages 1329–1338. ACM, 2014.

[70] Patrick E McSharry, Gari D Clifford, Lionel Tarassenko, et al. A dynamical model for generating synthetic electrocardiogram signals. *Biomedical Engineering*, 50(3):289–294, 2003.

[71] Stephen T Archer and Benjamin D Pless. Stimulation signal generator for an implantable device, 10th, Feb 2004. US Patent No. 6690974.

[72] John Chuang. One-step two-factor authentication with wearable biosensors. In *Symposium on Usable Privacy and Security-SOUPS*, volume 14, 2014.

[73] Tanvi Naik and Sheetal Koul. Multi-dimensional and multi-level authentication techniques. *International Journal of Computer Applications*, 75(12):17–22, 2013.

[74] Ivan Martinovic, Doug Davies, Mario Frank, Daniele Perito, et al. On the feasibility of side-channel attacks with brain-computer interfaces. In *Proceedings of the 21st USENIX conference on Security symposium*. USENIX Association, 2012.

[75] Tamara Bonaci, Ryan Calo, and Howard Jay Chizeck. App stores for the brain: Privacy & security in brain-computer interfaces. In *International Symposium on Science, Technology and Engineering*, pages 1–7, 2014.

[76] Howard Jay Chizeck and Tamara Bonaci. Brain-computer interface anonymizer, February 6 2014. US Patent App. 14/174,818.

[77] Lucas Davi, Alexandra Dmitrienko, Ahmad-Reza Sadeghi, and Marcel Winandy. Privilege escalation attacks on Android. pages 346–360, 2011.

[78] Earlence Fernandes, Bruno Crispo, and Mauro Conti. Fm 99.9, radio virus: Exploiting fm radio broadcasts for malware deployment. *IEEE Transactions on Information Forensics and Security*, 8(6):1027–1037, 2013.

[79] Mauro Conti, Nicola Dragoni, and Sebastiano Gottardo. Mithys: Mind the hand you shake-protecting mobile devices from ssl usage vulnerabilities. In *International Workshop on Security and Trust Management*, pages 65–81. Springer, 2013.

[80] William Enck, Peter Gilbert, Seungyeop Han, et al. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5, 2014.

[81] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, et al. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In *ACM SIGPLAN Notices*, pages 259–269, 2014.

[82] John R Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[83] Mauro Conti, Radha Poovendran, and Marco Secchiero. Fakebook: Detecting fake profiles in on-line social networks. In *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1071–1078. IEEE, 2012.

[84] Nora Jong. Why the number of people creating fake accounts and using second identity on facebook are increasing. `tinyurl.com/3uwq75x`, 2010.

[85] RIVA Richmond. Stolen facebook accounts for sale. *The New York Times*, 2, 2010.

[86] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 15–15. USENIX Association, 2012.

[87] Lei Jin, Hassan Takabi, and James BD Joshi. Towards active detection of identity clone attacks on online social networks. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 27–38. ACM, 2011.

[88] Aldert Vrij. A cognitive approach to lie detection. *Deception detection: Current challenges and new approaches*, pages 205–229, 2015.

[89] Jeffrey J Walczyk, Frank P Igou, Alexa P Dixon, and Talar Tcholakian. Advancing lie detection by inducing cognitive load on liars: a review of relevant theories and techniques guided by lessons from polygraph-based approaches. *Frontiers in psychology*, 4, 2013.

[90] Paul Chandler and John Sweller. Cognitive load theory and the format of instruction. *Cognition and instruction*, 8(4):293–332, 1991.

[91] Jeffrey J Walczyk, Karen S Roper, Eric Seemann, and Angela M Humphrey. Cognitive mechanisms underlying lying to questions: Response time as a cue to deception. *Applied Cognitive Psychology*, 17(7):755–774, 2003.

[92] Evelyne Debey, Jan De Houwer, and Bruno Verschuere. Lying relies on the truth. *Cognition*, 132(3):324–334, 2014.

[93] Victor A Gombos. The cognition of deception: the role of executive processes in producing lies. *Genetic, social, and general psychology monographs*, 132(3):197–214, 2006.

[94] Jennifer Vendemia, Robert F Buzan, and Stephanie L Simon-Dack. Reaction time of motor responses in two-stimulus paradigms involving deception and congruity with varying levels of difficulty. *Behavioural Neurology*, 16(1):25–36, 2005.

[95] Bruno Verschuere, Gershon Ben-Shakhar, and Ewout Meijer. *Memory detection: Theory and application of the Concealed Information Test.* Cambridge University Press, 2011.

[96] Giuseppe Sartori, Sara Agosta, Cristina Zogmaister, Santo Davide Ferrara, and Umberto Castiello. How to accurately detect autobiographical events. *Psychological science*, 19(8):772–780, 2008.

[97] Giuseppe Sartori, Graziella Orru, and Merylin Monaro. Detecting deception through kinematic analysis of hand movement. *International Journal of Psychophysiology*, (108):16, 2016.

[98] Merylin Monaro, Luciano Gamberini, and Giuseppe Sartori. The detection of faked identity using unexpected questions and mouse dynamics. *PloS one*, 12(5):e0177851, 2017.

[99] Merylin Monaro, Francesca Ileana Fugazza, Luciano Gamberini, and Giuseppe Sartori. How human-mouse interaction can accurately detect faked responses about identity. In *International Workshop on Symbiotic Interaction*, pages 115–124. Springer, 2016.

[100] Merylin Monaro, Luciano Gamberini, and Giuseppe Sartori. Identity verification using a kinematic memory detection technique. In *Advances in Neuroergonomics and Cognitive Engineering*, pages 123–132. Springer, 2017.

[101] Robert Moskovitch, Clint Feher, Arik Messerman, et al. Identity theft, computers and behavioral biometrics. In *International Conference on Intelligence and Security Informatics (ISI)*, pages 155–160. IEEE, 2009.

[102] Romain Giot, Mohamad El-Abed, and Christophe Rosenberger. Greyc keystroke: a benchmark for keystroke dynamics biometric systems. In *International Conference on Biometrics: Theory, Applications, and Systems (BTAS)*, pages 1–6. IEEE, 2009.

[103] R Stockton Gaines, William Lisowski, S James Press, and Norman Shapiro. Authentication by keystroke timing: Some preliminary results. Technical report, RAND CORP SANTA MONICA CA, 1980.

[104] Fabian Monrose and Aviel D Rubin. Keystroke dynamics as a biometric for authentication. *Future Generation computer systems*, 16(4):351–359, 2000.

[105] Mariusz Rybnik, Marek Tabedzki, and Khalid Saeed. A keystroke dynamics based system for user identification. In *Computer Information Systems and Industrial Management Applications*, pages 225–230. IEEE, 2008.

[106] Clayton Epp, Michael Lippold, and Regan L Mandryk. Identifying emotional states using keystroke dynamics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 715–724. ACM, 2011.

[107] Lisa M Vizer, Lina Zhou, and Andrew Sears. Automated stress detection using keystroke and linguistic features: An exploratory study. *International Journal of Human-Computer Studies*, 67(10):870–886, 2009.

[108] Rada Mihalcea and Carlo Strapparava. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP*, pages 309–312. Association for Computational Linguistics, 2009.

[109] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics, 2011.

[110] Lina Zhou. An empirical investigation of deception behavior in instant messaging. *IEEE Transactions on Professional Communication*, 48(2):147–160, 2005.

[111] Douglas C Derrick, Thomas O Meservy, Jeffrey L Jenkins, et al. Detecting deceptive chat-based communication using typing behavior and message cues. *ACM Transactions on Management Information Systems (TMIS)*, 4(2):9, 2013.

[112] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[113] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[114] Daniel L Schacter, C-Y Peter Chiu, and Kevin N Ochsner. Implicit memory: A selective review. *Annual review of neuroscience*, 16(1):159–182, 1993.

[115] Statista. Smartphone user penetration as percentage of total global population from 2014 to 2020, November 2016.

[116] Statista. Tablet user penetration worldwide as share of population from 2014 to 2020, June 2016.

[117] Xuetao Wei, Lorenzo Gomez, Iulian Neamtiu, and Michalis Faloutsos. Profiledroid: multi-layer profiling of android applications. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 137–148. ACM, 2012.

[118] John Tadrous and Ashutosh Sabharwal. Interactive app traffic: An action-based model and data-driven analysis. In *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 1–8. IEEE, 2016.

[119] Mikhail Afanasyev, Tsuwei Chen, Geoffrey M Voelker, and Alex C Snoeren. Usage patterns in an urban wifi network. *IEEE/ACM Transactions on Networking (TON)*, 18(5):1359–1372, 2010.

[120] Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 281–287. ACM, 2010.

[121] Nathaniel Husted and Steven Myers. Mobile location tracking in metro areas: malnets and others. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 85–96. ACM, 2010.

[122] Gregor Maier, Fabian Schneider, and Anja Feldmann. A first look at mobile hand-held device traffic. In *Passive and Active Measurement*, pages 161–170. Springer, 2010.

[123] Clayton Shepard, Ahmad Rahmati, Chad Tossell, Lin Zhong, and Phillip Kortum. Livelab: measuring wireless networks and smartphone users in the field. *ACM SIGMETRICS Performance Evaluation Review*, 38(3):15–20, 2011.

[124] Alessandro Finamore, Marco Mellia, Maurizio M Munafò, Ruben Torres, and Sanjay G Rao. Youtube everywhere: Impact of device and infrastructure synergies on user experience. In *Proceedings of the 2011 ACM SIGCOMM*, pages 345–360. ACM, 2011.

[125] Aaron Gember, Ashok Anand, and Aditya Akella. A comparative study of handheld and non-handheld traffic in campus wi-fi networks. In *Passive and Active Measurement*, pages 173–183. Springer, 2011.

[126] Sang-Woo Lee, Jun-Sang Park, Hyun-Shin Lee, and Myung-Sup Kim. A study on smart-phone traffic analysis. In *Network Operations and Management Symposium (APNOMS)*, pages 1–7. IEEE, 2011.

[127] Ashwin Rao, Arnaud Legout, Yeon-sup Lim, Don Towsley, Chadi Barakat, and Walid Dabbous. Network characteristics of video streaming traffic. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, page 25. ACM, 2011.

[128] Sudhir Kumar Baghel, Kirti Keshav, and Venkateswara Rao Manepalli. An investigation into traffic analysis for diverse data applications on smartphones. In *Communications (NCC)*, pages 1–5. IEEE, 2012.

[129] Xian Chen, Ruofan Jin, Kyoungwon Suh, Bing Wang, and Wei Wei. Network performance of smart mobile handhelds in a university cam-

pus wifi network. In *Proceedings of ACM conference on Internet measurement conference*, pages 315–328. ACM, 2012.

[130] Hyo-Sik Ham and Mi-Jung Choi. Application-level traffic analysis of smartphone users using embedded agents. In *Network Operations and Management Symposium (APNOMS)*, pages 1–4. IEEE, 2012.

[131] ABM Musa and Jakob Eriksson. Tracking unmodified smartphones using wi-fi monitors. In *Proceedings of the 10th ACM conference on embedded network sensor systems*, pages 281–294. ACM, 2012.

[132] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. andromaly: a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, 38(1):161–190, 2012.

[133] Ryan Stevens, Clint Gibler, Jon Crussell, Jeremy Erickson, and Hao Chen. Investigating user privacy in android ad libraries. In *Workshop on Mobile Security Technologies (MoST)*, volume 10, 2012.

[134] Xin Su, M Chuah, and Gang Tan. Smartphone dual defense protection framework: Detecting malicious applications in android markets. In *International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pages 153–160. IEEE, 2012.

[135] Te-En Wei, Ching-Hao Mao, Albert B Jeng, Hahn-Ming Lee, Horng-Tzer Wang, and Dong-Jie Wu. Android malware detection via a latent network behavior analysis. In *International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1251–1258. IEEE, 2012.

[136] Marco V Barbera, Alessandro Epasto, Alessandro Mei, Vasile C Perta, and Julinda Stefa. Signals from the crowd: uncovering social relationships through smartphone probes. In *Proceedings of the conference on Internet measurement conference*, pages 265–276. ACM, 2013.

[137] Hiroki Kuzuno and Satoshi Tonami. Signature generation for sensitive information leakage in android applications. In *International Conference on Data Engineering Workshops (ICDEW)*, pages 112–119. IEEE, 2013.

[138] Zafar Ayyub Qazi, Jeongkeun Lee, Tao Jin, Gowtham Bellala, Manfred Arndt, and Guevara Noubir. Application-awareness in sdn. *ACM SIGCOMM computer communication review*, 43(4):487–488, 2013.

[139] Ashwin Rao, A Molavi Kakhki, Abbas Razaghpanah, Amy Tang, Shen Wang, Justine Sherry, Phillipa Gill, Arvind Krishnamurthy, Arnaud

Legout, Alan Mislove, et al. Using the middle to meddle with mobile. *CCIS, Northeastern University, Tech. Rep., December*, 2013.

[140] Lanier Watkins, Cherita Corbett, Benjamin Salazar, Kevin Fairbanks, and William H Robinson. Using network traffic to remotely identify the type of applications executing on mobile devices. *Mobile Security Technologies (MoST)*, 2013.

[141] Yi-Chao Chen, Yong Liao, Mario Baldi, Sung-Ju Lee, and Lili Qiu. Os fingerprinting and tethering detection in mobile networks. In *Proceedings of Conference on Internet Measurement Conference*, pages 173–180. ACM, 2014.

[142] Scott E Coull and Kevin P Dyer. Traffic analysis of encrypted messaging services: Apple imessage and beyond. *ACM SIGCOMM Computer Communication Review*, 44(5):5–11, 2014.

[143] Jonathan Crussell, Ryan Stevens, and Hao Chen. Madfraud: Investigating ad fraud in android applications. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 123–134. ACM, 2014.

[144] Martina Lindorfer, Matthias Neugschwandtner, Lukas Weichselbaum, et al. Andrubis-1,000,000 apps later: A view on current android malware behaviors. In *Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, pages 3–17. IEEE, 2014.

[145] Asaf Shabtai, Lena Tenenboim-Chekina, Dudu Mimran, Lior Rokach, Bracha Shapira, and Yuval Elovici. Mobile malware detection through analysis of deviations in application network behavior. *Computers & Security*, 43:1–18, 2014.

[146] Nino Vincenzo Verde, Giuseppe Ateniese, Emanuele Gabrielli, et al. No nat'd user left behind: Fingerprinting users behind nat from netflow records alone. In *International Conference on Distributed Computing Systems (ICDCS)*, pages 218–227. IEEE, 2014.

[147] Zhenxiang Chen, Hongbo Han, Qiben Yan, Bo Yang, et al. A first look at android malware traffic in first few minutes. In *Trustcom/BigDataSE/ISPA*, volume 1, pages 206–213. IEEE, 2015.

[148] Kensuke Fukuda, Hirochika Asai, and Kenichi Nagami. Tracking the evolution and diversity in network usage of smartphones. In *Proceedings of the 2015 ACM Conference on Internet Measurement*, pages 253–266. ACM, 2015.

[149] Anh Le, Janus Varmarken, Simon Langhoff, Anastasia Shuba, et al. Antmonitor: A system for monitoring from mobile devices. In *Proceedings of the 2015 ACM SIGCOMM Workshop on Crowdsourcing and Crowdsharing of Big (Internet) Data*, pages 15–20. ACM, 2015.

[150] Kyungwon Park and Hyoungshick Kim. Encryption is not enough: Inferring user activities on kakaotalk with traffic analysis. In *International Workshop on Information Security Applications*, pages 254–265. Springer, 2015.

[151] Tapio Soikkeli and Antti Riikonen. Session level network usage patterns of mobile handsets. In *International Conference on Telecommunications (ConTEL)*, pages 1–8. IEEE, 2015.

[152] Yihang Song and Urs Hengartner. Privacyguard: A vpn-based platform to detect information leakage on android devices. In *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 15–26. ACM, 2015.

[153] Qinglong Wang, Amir Yahyavi, Bettina Kemme, and Wenbo He. I know what you did on your smartphone: Inferring app usage over encrypted data traffic. In *IEEE Conference on Communications and Network Security (CNS)*, pages 433–441. IEEE, 2015.

[154] Hongyi Yao, Gyan Ranjan, Alok Tongaonkar, Yong Liao, and Zhuoqing Morley Mao. Samples: Self adaptive mining of persistent lexical snippets for classifying mobile application traffic. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 439–451. ACM, 2015.

[155] Mehedee Zaman, Tazrian Siddiqui, Mohammad Rakib Amin, and Md Shohrab Hossain. Malware detection in android by network traffic analysis. In *International Conference on Networking Systems and Security (NSysS)*, pages 1–5. IEEE, 2015.

[156] Hasan Faik Alan and Jasleen Kaur. Can android applications be identified using only tcp/ip headers of their launch time traffic? In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 61–66. ACM, 2016.

[157] Sophon Mongkolluksamee, Vasaka Visoottiviseth, and Kensuke Fukuda. Combining communication patterns & traffic patterns to enhance mobile traffic identification performance. *Journal of Information Processing*, 24(2):247–254, 2016.

[158] Fairuz Amalina Narudin, Ali Feizollah, Nor Badrul Anuar, and Abdullah Gani. Evaluation of machine learning classifiers for mobile malware detection. *Soft Computing*, 20(1):343–357, 2016.

[159] Wipawee Nayam, Arguy Laolee, Luck Charoenwatana, and Kunwadee Sripanidkulchai. An analysis of mobile application network behavior. In *Proceedings of the 12th Asian Internet Engineering Conference*, pages 9–16. ACM, 2016.

[160] Jingjing Ren, Ashwin Rao, Martina Lindorfer, et al. Recon: Revealing and controlling PII leaks in mobile network traffic. In *Proceedings of the 14th Conference on Mobile Systems, Applications, and Services*, pages 361–374. ACM, 2016.

[161] Nicholas Ruffing, Ye Zhu, Rudy Libertini, Yong Guan, and Riccardo Bettati. Smartphone reconnaissance: operating system identification. In *Consumer Communications & Networking Conference (CCNC)*, pages 1086–1091. IEEE, 2016.

[162] Brendan Saltaformaggio, Hongjun Choi, Kristen Johnson, Yonghwi Kwon, Qi Zhang, Xiangyu Zhang, Dongyan Xu, and John Qian. Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic. In *WOOT*, 2016.

[163] Raphael Spreitzer, Simone Griesmayr, Thomas Korak, and Stefan Mangard. Exploiting data-usage statistics for website fingerprinting attacks on android. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 49–60. ACM, 2016.

[164] Eline Vanrykel, Gunes Acar, Michael Herrmann, and Claudia Diaz. Leaky birds: exploiting mobile application traffic for surveillance. In *International Conference on Financial Cryptography and Data Security*, pages 367–384. Springer, 2016.

[165] Shanshan Wang, Zhenxiang Chen, Lei Zhang, Qiben Yan, Bo Yang, Lizhi Peng, and Zhongtian Jia. Trafficav: An effective and explainable detection of mobile malware behavior using network traffic. In *24th International Symposium on Quality of Service (IWQoS)*, pages 1–6. IEEE, 2016.

[166] Anshul Arora and Sateesh K Peddoju. Minimizing network traffic features for android mobile malware detection. In *Proceedings of the 18th International Conference on Distributed Computing and Networking*, page 32. ACM, 2017.

[167] Andrea Continella, Yanick Fratantonio, Martina Lindorfer, et al. Obfuscation-resilient privacy leak detection for mobile apps through differential analysis. In *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS)*, pages 1–16, 2017.

[168] Ana Rosario Espada, María del Mar Gallardo, Alberto Salmerón, and Pedro Merino. Performance analysis of spotify® for android with model-based testing. *Mobile Information Systems*, 2017, 2017.

[169] Nikunj Malik, Jayanarayan Chandramouli, Prahlad Suresh, et al. Using network traffic to verify mobile device forensic artifacts. In *Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2017.

[170] Vincent F Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security*, 2017.

[171] Xuetao Wei, Nicholas C Valler, Harsha V Madhyastha, Iulian Neamtiu, and Michalis Faloutsos. Characterizing the behavior of handheld devices and its implications. *Computer Networks*, 114:1–12, 2017.

[172] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 255–263. ACM, 2006.

[173] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Proceedings of ACM workshop on Cloud computing security*, pages 31–42. ACM, 2009.

[174] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pages 103–114. ACM, 2011.

[175] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[176] Hyojoon Kim and Nick Feamster. Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119, 2013.

[177] Floodlight. `http://www.projectfloodlight.org/floodlight/`.

[178] POX/NOX. `http://sdnhub.org/tutorials/pox/`.

[179] OpenDaylight SDN Controller Platform Rest Reference. `https://wiki.opendaylight.org`.

[180] Myung-Ki Shin, Ki-Hyuk Nam, and Hyoung-Jun Kim. Software-defined networking (sdn): A reference architecture and open apis. In *International Conference on ICT Convergence (ICTC)*, pages 360–361. IEEE, 2012.

[181] Maciej Kuźniar, Peter Perešíni, and Dejan Kostić. What you need to know about sdn flow tables. In *International Conference on Passive and Active Network Measurement*, pages 347–359. Springer, 2015.

[182] Seungwon Shin, Phillip A Porras, Vinod Yegneswaran, Martin W Fong, Guofei Gu, and Mabry Tyson. Fresco: Modular composable security services for software-defined networks. In *NDSS*, 2013.

[183] Andreas Voellmy, Hyojoon Kim, and Nick Feamster. Procera: a language for high-level reactive network control. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 43–48. ACM, 2012.

[184] Matthew Monaco, Oliver Michel, and Eric Keller. Applying operating system principles to sdn controller design. In *Proceedings of the 12nd Workshop on Hot Topics in Networks*, page 2. ACM, 2013.

[185] Hesham Mekky, Fang Hao, Sarit Mukherjee, Zhi-Li Zhang, and TV Lakshman. Application-aware data plane processing in sdn. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 13–18. ACM, 2014.

[186] Philip Porras, Seungwon Shin, Vinod Yegneswaran, Martin Fong, Mabry Tyson, and Guofei Gu. A security enforcement kernel for openflow networks. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 121–126. ACM, 2012.

[187] Ehab S Al-Shaer and Hazem H Hamed. Discovery of policy anomalies in distributed firewalls. In *23rd Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2605–2616. IEEE, 2004.

[188] Adiseshu Hari, Subhash Suri, and Guru Parulkar. Detecting and resolving packet filter conflicts. In *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1203–1212. IEEE, 2000.

[189] Ehab Al-Shaer, Hazem Hamed, Raouf Boutaba, and Masum Hasan. Conflict classification and analysis of distributed firewall policies. *IEEE journal on selected areas in communications*, 23(10):2069–2084, 2005.

[190] Ferney A Maldonado-Lopez, Eusebi Calle, and Yezid Donoso. Detection and prevention of firewall-rule conflicts on software-defined networking. In *Reliable Networks Design and Modeling (RNDM)*, pages 259–265. IEEE, 2015.

[191] Ehab S Al-Shaer and Hazem H Hamed. Firewall policy advisor for anomaly discovery and rule editing. In *International Symposium on Integrated Network Management.*, pages 17–30. IEEE, 2003.

[192] OpenFlow Whitepaper for SDN. `http://archive.openflow.org/wp/tag/whitepaper/`.

[193] Ehab S Al-Shaer and Hazem H Hamed. Modeling and management of firewall policies. *IEEE Transactions on Network and Service Management*, 1(1):2–10, 2004.

[194] Ehab Al-Shaer and Hazem Hamed. Design and implementation of firewall policy advisor tools. *DePaul University, CTI, Tech. Rep*, 2002.

[195] Karamjeet Kaur, Japinder Singh, and Navtej Singh Ghumman. Mininet as software defined networking testing platform. In *International Conference on Communication, Computing & Systems (ICCCS*, 2014.

[196] Di Zhang, Zhenyu Zhou, Shahid Mumtaz, Jonathan Rodriguez, and Takuro Sato. One integrated energy efficiency proposal for 5G IoT communications. *IEEE Internet of Things Journal*, 3(6):1346–1354, 2016.

[197] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[198] SM Riazul Islam, Daehan Kwak, MD Humaun Kabir, Mahmud Hossain, and Kyung-Sup Kwak. The internet of things for health care: a comprehensive survey. *IEEE Access*, 3:678–708, 2015.

[199] Agusti Solanas, Constantinos Patsakis, Mauro Conti, Ioannis S Vlachos, Victoria Ramos, Francisco Falcone, Octavian Postolache, Pablo A Pérez-Martínez, Roberto Di Pietro, Despina N Perrea, et al. Smart health: a context-aware health paradigm within smart cities. *IEEE Communications Magazine*, 52(8):74–81, 2014.

[200] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[201] Qian Zhu, Ruicong Wang, Qi Chen, Yan Liu, and Weijun Qin. Iot gateway: Bridgingwireless sensor networks into internet of things. In *International Conference on Embedded and Ubiquitous Computing (EUC)*, pages 347–352. IEEE, 2010.

[202] An-Feng Liu, Xian-You Wu, Zhi-Gang Chen, and Wei-Hua Gui. Research on the energy hole problem based on unequal cluster-radius for wireless sensor networks. *Computer communications*, 33(3):302–321, 2010.

[203] Ashraf Wadaa, Stephan Olariu, Larry Wilson, Mohamed Eltoweissy, and K Jones. Training a wireless sensor network. *Mobile Networks and Applications*, 10(1-2):151–168, 2005.

[204] Xiaobing Wu, Guihai Chen, and Sajal K Das. Avoiding energy holes in wireless sensor networks with nonuniform node distribution. *IEEE Transactions on parallel and distributed systems*, 19(5):710–720, 2008.

[205] Abtin Keshavarzian, Huang Lee, and Lakshmi Venkatraman. Wakeup scheduling in wireless sensor networks. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 322–333. ACM, 2006.

[206] Maria Rita Palattella, Nicola Accettura, Luigi Alfredo Grieco, Gennaro Boggia, Mischa Dohler, and Thomas Engel. On optimal scheduling in duty-cycled industrial iot applications using IEEE802. 15.4 e TSCH. *IEEE Sensors Journal*, 13(10):3655–3666, 2013.

[207] Peng Lin, Chunming Qiao, and Xin Wang. Medium access control with a dynamic duty cycle for sensor networks. In *Wireless Communications and Networking Conference*, volume 3, pages 1534–1539. IEEE, 2004.

[208] Muralidhar Medidi and Yuanyuan Zhou. Extending lifetime with differential duty cycles in wireless sensor networks. In *Global Telecommunications Conference*, pages 1033–1037. IEEE, 2007.

[209] Stanislava Soro and Wendi B Heinzelman. Prolonging the lifetime of wireless sensor networks via unequal clustering. In *Parallel and Distributed Processing Symposium*, pages 8–pp. IEEE, 2005.

[210] Mei Yang, Shupeng Wang, Ahmed Abdelal, Yingtao Jiang, and Yoohwan Kim. An improved multi-layered architecture and its rotational scheme for large-scale wireless sensor networks. In *Consumer Communications and Networking Conference*, pages 855–859. IEEE, 2007.

[211] Vivek Mhatre and Catherine Rosenberg. Design guidelines for wireless sensor networks: communication, clustering and aggregation. *Ad hoc networks*, 2(1):45–63, 2004.

[212] Sean Dieter Tebje Kelly, Nagender Kumar Suryadevara, and Subhas Chandra Mukhopadhyay. Towards the implementation of iot for environmental condition monitoring in homes. *IEEE Sensors Journal*, 13(10):3846–3853, 2013.

[213] Stephan Olariu and Ivan Stojmenovic. Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting. In *INFOCOM 25th International Conference on Computer Communications.*, pages 1–12. IEEE, 2006.

[214] Chao Song, Ming Liu, Jiannong Cao, Yuan Zheng, Haigang Gong, and Guihai Chen. Maximizing network lifetime based on transmission range adjustment in wireless sensor networks. *Computer Communications*, 32(11):1316–1325, 2009.

[215] Wen-Hwa Liao, Yucheng Kao, and Ru-Ting Wu. Ant colony optimization based sensor deployment protocol for wireless sensor networks. *Expert Systems with Applications*, 38(6):6599–6605, 2011.

[216] T Baker, B Al-Dawsari, H Tawfik, D Reid, and Y Ngoko. Greedi: An energy efficient routing algorithm for big data on cloud. *Ad Hoc Networks*, 35:83–96, 2015.

[217] Thar Baker, Yanik Ngoko, Rafael Tolosana-Calasanz, Omer F Rana, and Martin Randles. Energy efficient cloud computing environment via autonomic meta-director framework. In *International Conference on Developments in eSystems Engineering (DeSE)*, pages 198–203. IEEE, 2013.

[218] Qiao-Qin Li, Haigang Gong, Ming Liu, Mei Yang, and Jun Zheng. On prolonging network lifetime through load-similar node deployment in wireless sensor networks. *Sensors*, 11(4):3527–3544, 2011.

[219] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, and Mani Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions on mobile computing*, 99(1):70–80, 2002.

[220] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd annual Hawaii international conference on System Sciences*, pages 10–pp. IEEE, 2000.

[221] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. 2014.

[222] Henri P Gavin and Jeffrey T Scruggs. Constrained optimization using lagrange multipliers. 2012.

[223] Dimosthenis Pediaditakis, Yuri Tselishchev, and Athanassios Boulis. Performance and scalability evaluation of the castalia wireless sensor network simulator. In *Proceedings of the 3rd international ICST conference on simulation tools and techniques*, page 53. ICST, 2010.

[224] Hai N Pham, Dimosthenis Pediaditakis, and Athanassios Boulis. From simulation to real deployments in wsn and back. In *International Conference on World of Wireless, Mobile and Multimedia Networks*, pages 1–6. IEEE, 2007.