An Adaptive Approach to Securing Ubiquitous Smart Devices in IoT Environment with Probabilistic User Behavior Prediction

by

Arun Balaji Buduru

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2016 by the
Graduate Supervisory Committee:

Sik-Sang Yau, Chair
Gail-Joon Ahn
Hasan Davulcu
Yanchao Zhang

ARIZONA STATE UNIVERSITY

December 2016

ABSTRACT

Cyber systems, including IoT (Internet of Things), are increasingly being used ubiquitously to vastly improve the efficiency and reduce the cost of critical application areas, such as finance, transportation, defense, and healthcare. Over the past two decades, computing efficiency and hardware cost have dramatically been improved. These improvements have made cyber systems omnipotent, and control many aspects of human lives. Emerging trends in successful cyber system breaches have shown increasing sophistication in attacks and that attackers are no longer limited by resources, including human and computing power. Most existing cyber defense systems for IoT systems have two major issues: (1) they do not incorporate human user behavior(s) and preferences in their approaches, and (2) they do not continuously learn from dynamic environment and effectively adapt to thwart sophisticated cyber-attacks. Consequently, the security solutions generated may not be usable or implementable by the user(s) thereby drastically reducing the effectiveness of these security solutions.

In order to address these major issues, a comprehensive approach to securing ubiquitous smart devices in IoT environment by incorporating probabilistic human user behavioral inputs is presented. The approach will include techniques to (1) protect the controller device(s) [smart phone or tablet] by continuously learning and authenticating the legitimate user based on the touch screen finger gestures in the background, without requiring users' to provide their finger gesture inputs intentionally for training purposes, and (2) efficiently configure IoT devices through controller device(s), in conformance with the probabilistic human user behavior(s) and preferences, to effectively adapt IoT devices to the changing environment. The effectiveness of the approach will be demonstrated with experiments that are based on collected user behavioral data and simulations.

Index terms- pro-active protection, predictive defense, probabilistic reasoning, continuous authentication, dynamic IoT environment assessment and probabilistic human behaviors

ACKNOWLEDGMENTS

Being a PhD student in the Computer Science Program in the School of Computing, Informatics and Decision Systems Engineering, and in the Information Assurance Center at Arizona State University was lots of fun. Joining it was one of the best decisions that I have ever made. I want to thank my adviser, Stephen S. Yau. He is an excellent mentor who gave me the freedom and the encouragement to explore interdisciplinary research ideas. I am grateful for having been his student. He is a source of inspiration to all students. I would also like to thank my committee members Gail-Joon Ahn, Hasan Davulcu and Yanchao Zhang for their helpful suggestions and insightful comments.

Special thanks to Yaozhong Song, Vinjith Nagaraja, Sayantan Guha, David Lucero, Jeremy Wright, Vishnu Teja Kilari, Jiayu Zhou, Raymond Tu, Scott Menor and other members at Arizona State University for their invaluable inputs and fiery discussions. I would also like to thank Michael Freed of Cisco’s IoT Architecture research lab and members of Qualcomm research lab for their mentorship and support.

But most of all, I want to express the deepest gratitude to my parents, Divakar Buduru and Suganya Pappian, for their unwavering confidence and support. Special thanks to my brother Praveen Balaji Buduru. I dedicate this dissertation work to my parents.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Overview

Ubiquitous cyber systems, including Internet of Things (IoT) systems are being increasingly used to vastly improve efficiencies and reduce costs in areas such as finance, health care, power management, transportation, and defense. Over the past twenty years computing efficiency and hardware costs have dramatically improved, these have made the cyber and IoT systems omnipotent, and touch all aspects of human lives. Recent surveys have shown that the growth of the ubiquitous smart devices will accelerate in the near future [1, 2]. Past trends have shown that any widely used entity becomes a target for attacks since potential system breaches offer huge incentives to attackers. Extensive research has been done in developing cyber defense systems that provide necessary defensive and offensive measures to mitigate the effects of the attacks [3, 4]. But recent security breaches in cyber systems used in finance, defense and even nuclear plants have shown that current cyber defense mechanisms are unable to cope with these new breed of attacks [5]. The integrity and security of these cyber systems are not only vital to those respective areas but also have a vast impact on the safety of humans.

One of the major deficiencies in developing comprehensive adaptive cyber defense systems, including ones for IoT, is that most of the solutions do not include human behavior and preferences in cyber defense approaches [3, 6]. Here, the human can be either the person monitoring the cyber defense system or the attacker launching the attacks. The two main reasons for the need to include the human behavior(s) in securing ubiquitous smart devices in IoT environment are (1) to effectively adapt smart devices in accordance with the user(s) behavior and preferences, and (2) to improve the usability of the cyber defense solutions.

A few of the most commonly used devices to control the other devices in an IoT application environment are the smart devices such as smart phone, tablet. These smart devices are traditionally protected through PIN, pass phrase, pattern, face recognition and recently fingerprint authentication [7]. One of the major weaknesses of these authentication factors is that they do not continuously authenticate users after the initial authentication [8]. Hence, if the smart device is physically compromised (such as being stolen) after the authentication has been performed, then the attacker can keep the authenticated session open and extract the confidential data before the session times out, or the user shuts down the device through other means. The small size of smart devices makes them particularly vulnerable for theft. Hence it is necessary to develop an approach to continuously authenticating the legitimate user on the smart devices without requiring frequent cognizable inputs.

Some of the approaches use techniques such as game theory to develop pro-active cyber defense techniques [9, 10], but they have major limitations due to the assumptions used in these approaches, such as the rationality and Nash equilibrium, which may not be valid and/or feasible for current and emerging cyber infrastructure. As a consequence of not including human behavior, the cyber defense systems, including those for IoT applications, are either not dynamic enough to thwart sophisticated cyber-attacks [11] or the restrictive/intrusive nature of cyber defense system will make it very difficult to effectively use them in real world applications. Given the circumstances of increasing sophistication of cyber-attacks there is an urgent need to develop a user centric approach to securing IoT through probabilistic human behavior prediction.

In order to address these security concerns we develop a comprehensive user-centric approach to securing IoT systems by predicting user behavior(s). In this dissertation work we will be focusing on two of the following major components of our comprehensive approach: (1) protecting the controller device through a continuous authentication approach to continuously authenticate the legitimate user usage session based on the legitimate user touch screen finger gestures, (2) planning various device actions through a controller device [smart phone (or) tablet] to satisfy user requirements securely and efficiently in IoT applications by predicting the human behavior. We will be using existing approaches to configuring IoT application devices, which is the third component of our overall approach. The overall theme of this dissertation work will be to incorporate human behavioral traits into security solutions to enable effective adaptation to the dynamically changing users' environment.

First, we develop an adaptive approach for continuously authenticating legitimate user on a touch screen based smart device (or controller device). The specific goals of our approach [12] are: (1) continuously authenticate user on a touch based smart device without requiring frequent cognizable user input, (2) adapt the authentication algorithm efficiently based on the changing legitimate user behavior with minimal resources, and (3) leverage the probabilistic nature of user behavior to improve the efficiency of the continuous authentication. Our approach includes the following algorithms: 1) an algorithm to estimate the context of smart device usage, 2) an algorithm to estimate degree of importance of each cell/state on the user's smart device touch screen, and 3) a technique that uses Markov Decision Process (MDP) algorithm to continuously authenticate users' with computed user gesture model. Each of the above items will be discussed in detail in the Chapter 4.

Second, we develop an effective approach for intelligent planning of device action in IoT applications. Our approach [13] is designed to work for both individual user and users as a group. The specific goals of our approach are: (1) assist the user(s) of the IoT application to efficiently secure their environments, (2) observe and monitor user(s) behavior to plan device actions that complement the user behavior and preferences, and (3) ensure that the user requirements are satisfied with minimal resources. Our approach includes the following algorithms: 1) a state analysis technique for the service provider, 2) learning algorithm for dynamic IoT application assessment, and 3) a technique that uses MDP (Markov Decision Process) planning to generate efficient IoT device action plans. Each of the above items will be discussed in detail in the Chapter 5.

1.2 Organization of Dissertation

The dissertation work is organized as follows: the current state of the art related to my research will be presented in Chapter 2. Overall approach to dynamically secure ubiquitous devices with probabilistic human behavior modelling will be presented in Chapter 3. A reinforcement learning based approach to continuously authenticate users' finger gestures using touch-based smart phone without requiring users’ to provide intentional finger gestures for training purposes, including the experimental results, will be presented in Chapter 4. An approach to adaptively reconfigure the smart devices in an IoT environment based on users’ behavioral preferences and requirements, including the simulation results, will be presented in Chapter 5. The conclusion of this dissertation work and future research directions will be presented in Chapter 6.

Chapter 2

CURRENT STATE OF ART

In this chapter, the background information and existing research efforts related to the intelligent planning in IoT applications and continuous user authentication are discussed. Most of the cyber defense approaches [6, 14] do not include human behavior because it introduces a lot of uncertainties, which increases the complexity of the approaches. Intrinsically it is very hard to incorporate the human behavior into computing cyber defense paradigm because major human behavioral features are (1) probabilistic in nature, (2) depends on the context of the environment, and (3) continuously changes with the passage of time. In the absence of effective framework or techniques, each of these above features add a large overhead for including these features as a part of a cyber-defense system.

A few of the current approaches [14-17] to secure the ubiquitous smart devices in the IoT environment include the human behavior(s), but they make the assumption that the human behavior(s) to be deterministic. The deterministic assumption vastly limits the feasibility of their approached since they do not accurately capture the human behavior in the real world as they are probabilistic in nature. In the below subsection we will discuss the current state of the art in the specific application area.

2.1 Continuous User Authentication

Most of the current smart devices use the authentication factors to authenticate the legitimate user of a smart device, and these authentication factors have been generally classified into three categories: what you know, what you have and what you are [7]. The authentication factors belonging to the categories of what you know and what you have are not suitable for continuous user authentication of smart devices because the user needs to frequently provide conscious inputs for continuous authentication, which will likely cause undesirable user experience. This leaves us with only the category of what you are. Even in this category, some factors, such as fingerprints, iris and face recognition, are inherently infeasible due to the requirements of additional hardware in low-cost touch-screen smart devices, and frequent conscious user interactions. Furthermore, existing approaches [18-23] to continuous user authentication use centralized architecture, where a server collects specific user data for a specific time interval in the training phase, generates the authentication model from the user data, and uses an authentication model to authenticate the user during subsequent usage. If the user behavior changes, then the user needs to undergo the training phase again so that the authentication model for the user is updated with the latest information of the user's behavior. Hence, this process has inherent security, privacy, overhead and scalability problems

Password authentication schemes fall into the *what you know category*, and the hardware dongles, like RSA SecurId, fall into the what you have category. Biometric authentication schemes, such as fingerprint, iris and facepattern recognition, fall into the *what you are category*. The recent trend of authentication systems involves two-factor authentications, in which the password is usually accompanied by a one-time sign in code sent to the smart device. This trend combines the two authentication factors, what you know (password) and what you have (mobile device), to provide a slightly better authentication. The inherent problem with this two-factor authentication system is that in each of these two factors, sensitive information is lost in the event of theft of the smart device or attacker(s) breaking the two-factor authentication scheme. Continuous user authentication schemes using the authentication factor “What you are” can be broadly categorized into two groups: using touch gestures [20, 21] or multi-modality classification focusing on the data from a variety of sensors to model the user behavior [22, 23]. Most of the continuous authentication techniques in both categories involve an enrollment (training) phase and an authentication phase. The system learns the necessary patterns from the user’s related biological and/or sensory data, and stores them in the enrollment phase. In the authentication phase, the system compares the observed user related biological and/or sensory data against the patterns stored during the enrollment phase to re-authenticate a user.

Niu and Hao [18] conducted continuous user authentication experiments on iPad using user behavioral features in multitouch operations, involving 34 volunteers. Their experiments showed Equal Error Rates (EERs) of 7%-15% for one mode of multi-touch and EERs of 2.6%-3.9% if two multi-touch modes are combined. Frank, et al, [19] presented an approach to continuous user authentication based on their touch gestures with EERs in the range up to 4% using a kNearest Neighbors (k-NN) classifier and a Support Vector Machine (SVM) classifier. They demonstrated this approach to extract 30 features from a dataset of 41 users based on users’ swipe/scroll gestures. Later, Li, et al,

[20] presented a touch-based authentication technique on Motorola Droid phones running Android 2.2. This approach leverages the device files being used by Linux Multi touch protocol to monitor all the touch gestures of the user on the smart phone. This method was evaluated using SVM on 75 users who were allowed to use the smart phones freely, and showed the average classification accuracies of 95%.

For the approaches combining multiple biometric inputs to produce aggregated user identification results, Muncaster and Turk [21] presented an approach to performing continuous, score-level multi-modal authentication based on a weighted sum of scores from each modality. A continuous multi-modal biometrics system using a hidden Markov model (HMM) was developed by Sim, et al, [22]. It includes integration of the results from a fingerprint biometric classifier with a face classifier to improve the accuracy of continuous user authentication. Shi, at el, [23] used multimodal inputs, such as voice, location, multi-touch and motion, to perform continuous user authentication. Using Naïve Bayes classifiers on the features of multi-modal inputs, they showed average accuracy of over 95%.

All the above techniques are based on the assumption of zero-effort threats, where the adversary is assumed to be incapable of pulling off advanced forgery attacks [11] on the system, which are based on the characteristics of the user behavioral biometric patterns exhibiting large intra-user variation and overlap across a large population. The attackers leverage these characteristics to extract users' statistics from a large population database and perform advanced forgery attacks which can defeat these authentication techniques. The attack technique using robotic arm [11] increases the EERs in the continuous authentication techniques. In order to thwart these kinds of attacks, it is required to capture more unique characteristics of the user's finger gestures for differentiating the legitimate user from others. This increases the effort required on the part of the attackers to learn the user's behavior from a large population database, which may act as an effective deterrence.

Based on the above discussion, the existing continuous user authentication techniques fail to capture the nuances of the user's gestures, and they can only capture the user's gestures during the training phase. This will lead the continuous authentication techniques to miss some important strains of the user's behavior exhibited outside the training phase. This is particularly dangerous because, if the user model does not capture the unique characteristics of the user gestures, an attacker with access to a large set of normal user data might be able to construct a model conforming to the characteristics of the majority of the features exhibited by the population. This will enable the attacker to break the continuous authentication techniques. In order to capture a majority of unique characteristics of the user's gestures on a smart device, the technique needs to incorporate continuous learning of the user's gestures. In our approach, we will use a continuous authentication technique, which can continuously update the user's finger gesture model to effectively authenticate the legitimate user

2.2 Intelligent Device Adaptation in IoT Environments

In developing an intelligent mobile IoT application, it is important to understand that almost all u-things have serious limitations on processing power, and any additional computational load above and beyond their functional requirements, such as situation awareness, active interactions among other u-things in their environments, will cause serious degradation of the performance of the u-things which will likely lead to users' dissatisfaction.

In [24-26], the approaches to developing smart environments (same as mobile intelligent IoT application environments) assume that each u-thing has situation-aware capability to generate decisions specific to that autonomous u-thing based on its environmental situation. But, these approaches may lead to severe performance degradation of most of the u-things because not all u-things are designed and/or have the capability to handle such computationally very expensive operations.

In [27], a 3-layer architecture for developing smart environments was presented. In this approach, there is no information on how situational context models of the users' environments are generated. In [28], an approach was presented to developing smart environments, where the users need to setup and store their personal preferences on an NFC (Near Field Communication) enabled smart phone. Based on the preferences, the NFC enabled smart phone will control the other u-things to achieve the goals of the users' applications. However, this approach is unrealistic because the users are expected to know best possible modes of operation for all the devices' in the application environment.

In [29], an ontology-based activity recognition and scalable hierarchical planning technique is presented to maximize the energy efficiency in a smart office building. But, even if it is possible to generate "ontological model activities" for all possible user activities, it is not feasible to perform comparative selection of the ontological model activities for recognizing the user activities in real time which are essential inputs for planning algorithms. In [30], a decentralized control and decision system was presented for smart buildings, but there is no adequate explanation on how the decision making manager is trained and/or equipped to make smart decisions on the actions needed to be taken to improve the energy efficiency of the user environment.

Most of these above approaches [24-27] were under the implicit assumption that all the devices in an environment are smart devices and can communicate with other smart devices with ease. This assumption may not valid since not all the devices in an intelligent mobile IoT application environment can communicate on a peer-to-peer network architecture. Hence we developed our approach such that a u-thing needs to have communication link with only one of the control modules (CMs) of its application environment and the u-things is not required to do expensive computations as most of it is done either on one of CMs or mobile cloud

Chapter 3

OVERALL APPROACH TO SECURING UBIQUITOUS DEVICES

The primary focus of our approach is to developing a comprehensive user centric approach for securing ubiquitous devices in Internet of Things (IoT) environment through Probabilistic Human Behavior Prediction. In our approach we define human behavior(s) as a structured sequence of context-sensitive decisions [31]. In our approach we make three basic assumptions, which are, (1) human behavior(s) are intrinsically probabilistic in nature, (2) human behavior(s) follow the Markovian property, and (3) human behavior(s) is dynamic and evolves with the passage of time. We will, in this section, justify the feasibility of our assumptions.

Several researches [32, 33] in the human behavior(s) analysis have shown that human behavior(s) can be best modelled with the semi-deterministic systems using probabilities. This is because the semi-deterministic decision making system generate the decisions based on the various contextual parameters and forcing parameters, which can include environmental factors and historical behavioral preferences. The human behavioral decision making systems also follow similar decision making process [34 - 36], thus semi-deterministic decision making system ideal to model the human decision making systems.

For single agent decision systems, Markovian property [37] states that any decision taken by the agent is only dependent on the parameter of the current state and is independent of parameters of all its previous states. We in our approach assume that the human decision making generally conform to the Markovian property. This is because majority of the human behavior(s) generally tend to be highly contextual in nature, meaning they are heavily dependent on the current state or context [31]. Note that we do not claim that all the human behavior(s) conform to Markovian property. But we do assume that the human behavior(s) we consider in our approach, specifically human finger gestures on smart device touch screens and contextual human behavior(s), which are highly contextual in nature, to conform the Markovian property.

Human behavior(s) generally tend to be dynamic in nature since the parameters of the environment and the forcing functions tend to be dynamic in nature. Since most of the human decision making is highly contextual the human behavior(s) also tend to be dynamic in nature. Hence our third assumption is valid and feasible.

The primary goal of our approach is to develop a mathematical framework for efficiently incorporating the three major human behavioral characteristics, which are: (1) human behavior is probabilistic in nature, (2) human behavior depends on the context of the surrounding environment, and (3) human behavior continuously changes with the passage of time. Our user centric approach to securing ubiquitous devices has three major components as shown in Figure 1. The first component continuously learns and authenticates the legitimate user. The second component describes the intelligent adaptation of IoT device states. The IoT device states' adaptation is complex since it is dynamic and dependent on many environmental variables. Hence our approach also takes into to account the users' environment requirements, dynamic IoT environment and users' usage patterns and preferences to effectively adapt the IoT device states.

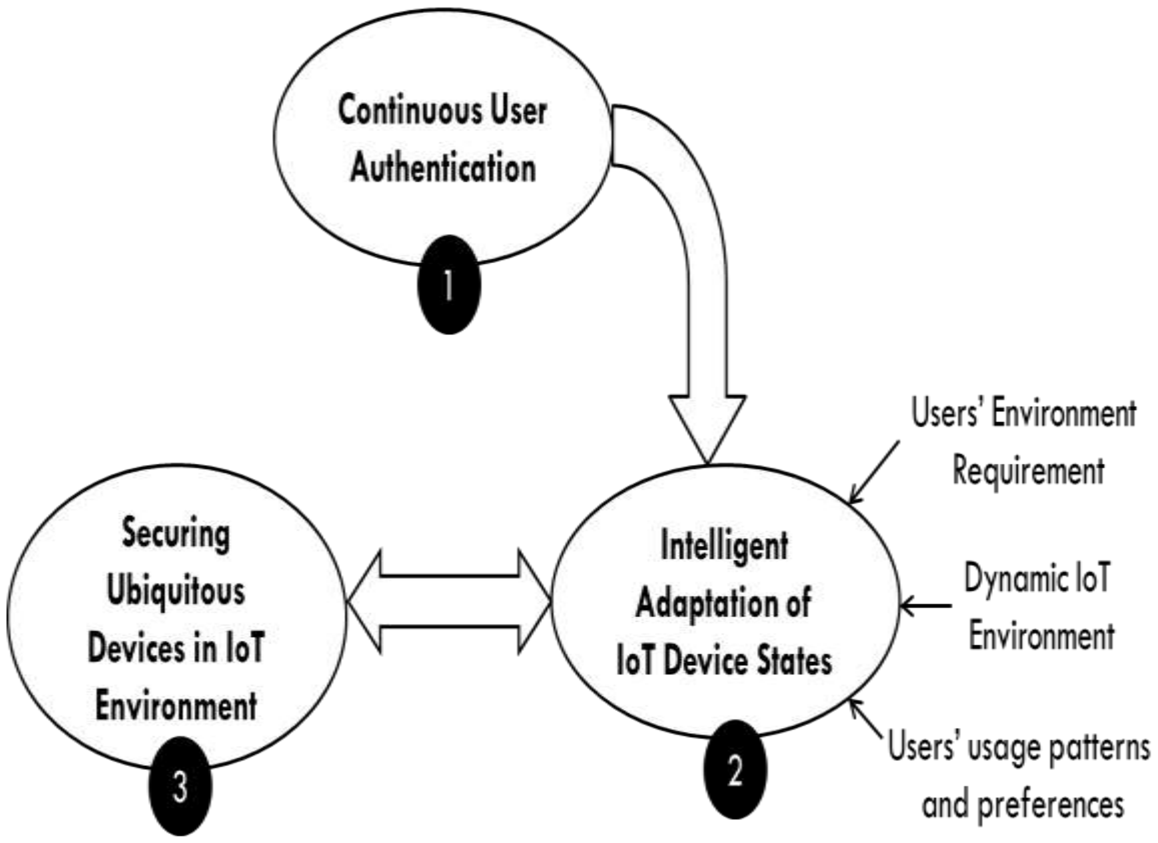

Continuous User Authentication
1
Users' Environment Requirement
Securing Ubiquitous Devices in IoT Environment
Intelligent Adaptation of IoT Device States
Dynamic IoT Environment
3
2
Users' usage patterns and preferences

Figure 1: System Diagram of Our Overall Approach to Securing Ubiquitous Devices in IoT Environment

The following are the steps of our overall approach,

Step 1) Continuously authenticate the legitimate user through an adaptive continuous authentication on controller device [smart phone (or) tablet]. This is essential since the controller is in direct contact with the user(s) and has the ability to manipulate the device actions. More details will be presented in Chapter 4.

Step 2) Intelligent adaptation of various devices through a controller device [smart phone (or) tablet] to satisfy user requirements securely and efficiently in IoT applications. More details about intelligent planning are presented in Chapter 5.

Step 3) Based on the plans generated in Step 2) the individual devices in the IoT application environment can be configured and secured through the controller device. We will use the existing approaches to efficiently actuate the device configuration updates.

In order to predict the human behavior(s) we need to develop an approach that accurately models the human decision making process. This requires a detailed understanding on the operational aspects of the human decision making system. Based on the extensive research in human psychology and economics it has been established that the human brain uses dual decision system to process information [38] and generate decisions in the real world. The human dual decision system is temporal in nature and has two decision generation techniques, (1) short horizon decisions, for quicker decision making, and (2) long horizon decisions, for slower decision making. The dual decision system provides a promising blueprint for developing an effective approach for accurately capturing human behavior(s). Most of the human decisions are short horizon decisions. It is estimated that humans make tens of thousands of short horizon decisions every day [38]. On the other hand the long horizon decisions are used fewer times comparatively.

In short horizon, the time window is a few seconds and is similar to the fast decision making or “intuitive” human thinking. In long horizon, the time window is a few hours/days and is similar to slow decision making or “reasoning” based human thinking

[38]. It is also worth noting that the long horizon decision making also involves planning and simulation since it requires anticipation of the actions of various actors in the environment. The short and the long horizon dual decision system is extremely important to the effectively securing smart devices since this enables us to flexibly apply discriminatory and as well as reason based decision making depending on the dynamic IoT environment.

In addition, extensive research in human psychology has shown that humans have around hundreds of cognitive biases [39]. These cognitive biases have varying degree of influence over the dual decision making system. Since the short horizon decision making is quick the cognitive biases tend to affect the short horizon decision making than the long horizon decision making process. The long horizon decision making process usually involves simulation and planning of the future set of actions, hence the influence of the cognitive biases may be reduced due to the increased exercise of cognitive thinking.

Our approach is designed to emulate and accommodate the human dual decision making process and probabilistic characteristics of human behaviors. Our approach will include the mixture model containing the semi-supervised learners and the MDP networks that will generate “reasoning” based decisions for long horizon scenarios and the semi-supervised learners will generate “intuitive” decisions for short horizon scenarios. The rationale behind the utilization of the semi-supervised learners is to reduce the usage of the domain experts. Traditional supervised based learners rely on labeled training data, which is generated by domain experts manually, in order to effectively train the model for the assessment algorithms. The broader idea in our approach is to incorporate human-like adaptive intelligence into the systems to enable them take both quick decisions and reasoned decisions, as required, at the same time reducing the resource utilization.

Chapter 4

CONTINOUS USER AUTHENTICAITON IN TOUCH SCREEN SMART DEVICES

Smart devices are experiencing explosive growth in terms of the number in use as well as the services offered by vendors. According to a recent report [1, 2] by Gartner, vendors shipped about 2.5 billion touch-screen smart devices in 2014 and shipments are projected to reach 3 billion by the end of 2017. The information available in personal smart devices, such as smartphones and tablets, includes more detailed and precise information on their users' personal attributes than the information in other computing devices since personal smart devices are rarely shared among different users. Proliferation of applications across various domains, such as banking, shopping, and healthcare, requires smart devices to store even more sensitive user data. Furthermore, the rising BYOD (Bring Your Own Device) trend is leading to storage of enterprise data on personal smart devices [40]. The increasing number of sensors in smart devices makes them sources of rich personal data of the users, like frequently visited locations. All these trends make personal smart devices more attractive targets for malicious activities, ranging from physically stolen devices to infected smart devices by malware.

Smart devices are traditionally protected by authentication methods using PIN, passphrase, pattern, face recognition and fingerprint [7]. One of the major weaknesses of these authentication methods is that they do not continuously authenticate users after the initial authentication. Hence, if a smart device is physically compromised (including being stolen) after the authentication has been performed, the attacker can keep the authenticated session open and extract the confidential data before the session times out or the user shuts down the device. Smart devices are usually small and make them particularly vulnerable for theft. Although there are various ways to remotely locate, erase and lock a smart device in case of theft, all of these mechanisms can be circumvented by methods, like removing the SIM card and disconnecting access to internet. Smartphones pose additional challenges to the existing authentication mechanisms, such as having small screen size and computational resources. Small screen size prevents users from using “strong” passwords because it normally requires a mixture of letters, numbers and symbols which are not present in existing smart device lock screen.

Recent successful breaches into the smart devices have shown that the existing numerical or pattern based authentication systems are not adequate. Existing software, such as ElcomSoft, can be used to copy the encrypted filesystem and perform brute force attacks. Additionally, the numerical authentication pins are insufficient as 4-digit and 8-digit PINs have been shown to be broken within 40 minutes and 4months respectively [8]. Furthermore, current state of the art fingerprint based authentication systems are even more dangerous as the user don’t even have to be conscious to authenticate the usage session.

To address these issues, a mechanism, that authenticates the user continuously through the entire usage session which is unobservable to the user, is needed. In this chapter, we will present an approach using Markov decision process (MDP) and touch gestures of the user to continuously authenticate the user on a touch-screen smart device. This approach will include algorithms to estimate the degree of importance of each cell on the touch screen of the user's smart device for continuous authentication, and to identify the usage context of the smart device. This approach has two major advantages. One is that this approach is more efficient by adaptively updating the authentication model with evolving user's finger gestures. The other major advantage is to have better authentication accuracy by treating uninterrupted user finger gestures over a short time interval as a single gesture for continuous user authentication.

The organization of this chapter is as follows: After discussing the overall approach in Section 4.1, the features used for continuously authenticating a smart device's user will be presented in Section 4.2. A method of estimating the degree of importance of each cell on the touch screen of the user's smart device to continuous user authentication will be presented in Section 4.3. The estimation of usage context of the smart device based on user's touch gestures will be presented in Section 4.4. In Section 4.5, we will present a technique using the MDP for continuous user authentication. An example to illustrate a part of our approach will be given in Section 4.6, and conclusions and future research will be discussed in Section 4.7.

The research tasks proposed in this section includes developing approach that uses MDP for performing adaptive continuous authentication of legitimate users on touch based smart devices. The specific goals of our approach are (1) continuously authenticate user on a touch based smart device without requiring frequent cognizable user input, (2) adapt the authentication algorithm efficiently based on the changing legitimate user behavior with minimal resources, and (3) leverage the probabilistic nature of user behavior to improve the efficiency of the continuous authentication.

4.1 Approach

In this section we will present our approach algorithm to continuously authenticate the user on a touch-based smart device. This approach will include an algorithm to estimate the degree of importance of each cell/state on the user's smart device touch screen, estimate the context in which the smart device is being used by the user, and a technique that uses the Markov decision process (MDP) to continuously authenticate legitimate user. In our approach, we assume that all the states of the smart device are known and observable. This is a valid assumption since in our approach the observable smart device screen is divided into smaller cells and each cell represents a state in the MDP state graph. A sequence of states represents a user gesture. Since all the cells are known and the structure of the grid is well defined, the MDP can be applied to our problem. Our approach has the following features: backward compatible with older versions of smart devices, lightweight, and easy to use. In our approach we assume that an adversary has the following capabilities: (1) able to observe the behavior of the user on the smart device (shoulder surfing), (2) able to physically steal user's smart device, and (3) knowledgeable of the user's smart device authentication pin.

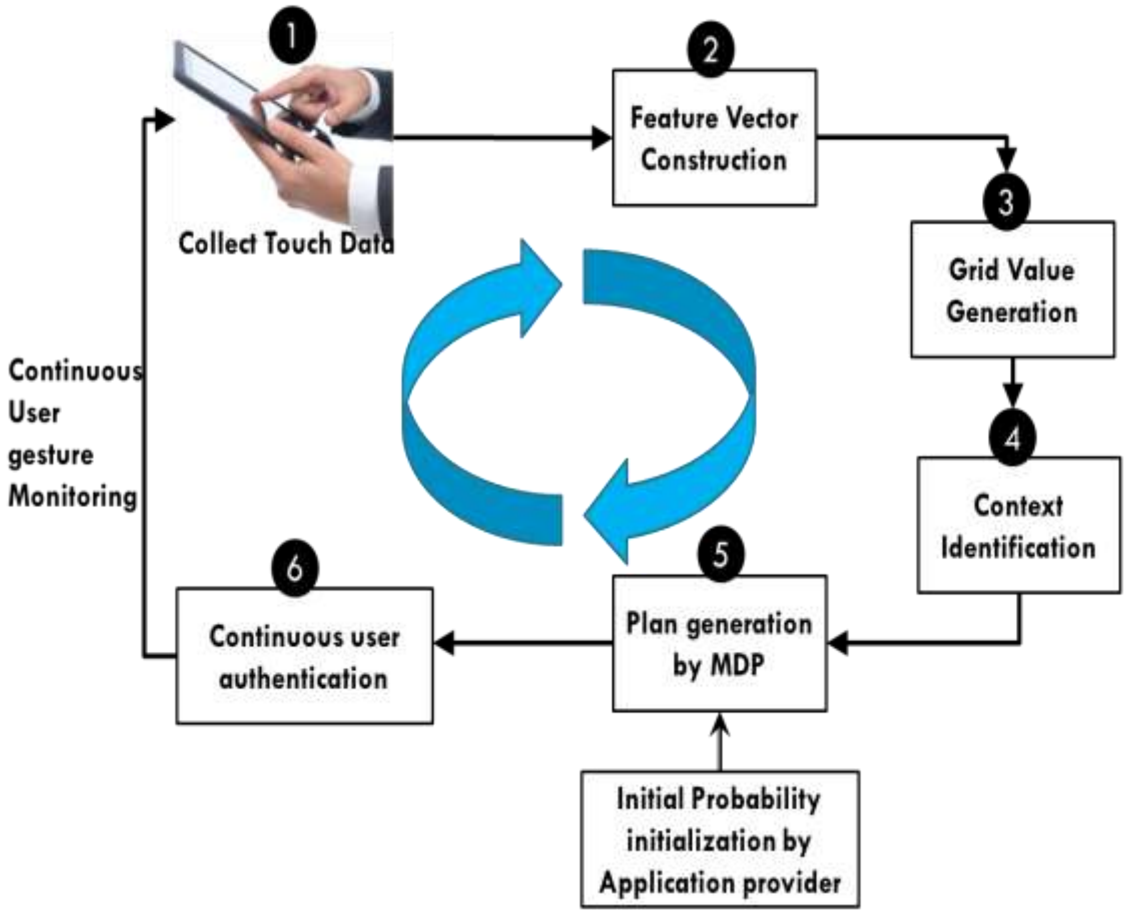

1
Collect Touch Data
2
Feature Vector Construction
3
Grid Value Generation
4
Context Identification
5
Plan generation by MDP
6
Continuous user authentication
Initial Probability initialization by Application provider
Continuous User gesture Monitoring

Figure 2: System Diagram for Our Approach for Continuous User Authentication

In our approach, as shown in Figure 2, we will define two different kinds of user gestures; one is micro-gesture which is defined as uninterrupted user finger movement on the smart device screen, and the other is the macro-gesture which is defined is as collection of micro-gestures over a preset period of time. This preset period also will define the frequency of the continuous authentication of the user. We will use a data acquisition procedure for generating the grid value which is the estimated degree of importance of each cell/state on the user's smart device touch screen.

Our overall approach [12] is shown in Figure 2, and its major steps are designated by the numbers in the figure,

Step 1) Continuously collect the sensory data from the user's smart device when the user is using the smart device. Since most of the features we require are similar to those used in [41], we use the same procedure in [41] to extract the data from the mobile device.

Step 2) Analyze the sensory data flows collected during the user's usage session of the smart device in Step 1 to extract the necessary information to construct the required features. The output of this step is the features which are used for grid value generation (GVG) process.

Step 3) Using the features constructed in Step 2 to run the GVG algorithm on user's smart device for continuously updating the MDP state graph during the user's smart device usage session. The inputs for the GVG algorithm include the attributes of user gestures on the smart device screen, such as pressure, speed and usage frequency.

Step 4) Identify the context in which the touch screen of the smart device is being used by the legitimate user using the primary and secondary features.

Step 5) Generate the plans which are the sequences of cells/states in the MDP state graph representing the user's gestures, along with the plans' respective cumulative reward value. These plans are generated using our technique that uses the MDP algorithm.

Step 6) The cumulative reward values of the plans are then used as benchmark data to continuously authenticate the legitimate user of the smart device. The cumulative reward values of the plans are compared with the cumulative reward value for user's touch gesture for effective authentication.

Our continuous authentication approach is flexible to the selection of number of cells and can be selected based on factors, such as type of user's smart device, screen size, degree of uniqueness of user gestures to be captured. It is also important to note that increasing the number of cells also increases the operational complexity. So an optimal trade-off is needed to be achieved based on the application specification, since the accuracy of the continuous authentication depends on the number of cells on touch screen of the smart device accurately. The details of the grid representation in the MDP will be explained in Section 4.3.

4.2 Representation of user finger gestures

In our approach, we use two kinds of user gestures; one is micro-gesture which is uninterrupted user's finger movement on the smart device screen, and the other is the macro-gesture which is a collection of micro-gestures over a preset time interval. We will use the data acquisition procedure [41] for extracting raw user's finger gesture data from the touch screen of the smart device.

In order to efficiently represent the user's finger gestures on the touch screen of the smart device, the set of features needs to be simple, and yet can represent the various finger gestures reflecting the user's unique usage characteristics of the smart device. This will ensure that the collected information from the touch screen of the smart device is sufficient to distinguish the legitimate user of a smart device from other users, including malicious users. In our approach, we select dynamic and unique parameters, such as user's finger touch pressure and speed of user's finger gestures. We set two levels of features, called primarily and secondary, from the features used in the smart device authentication techniques [41] to ensure the efficient representation of user's finger gestures and accurate continuous authentication of the legitimate user. Our illustration in Section 4.7 shows that these features are reasonably good and sufficient for continuous user authentication.

1) Primary features:

   a) First touch (FT) of a cell, represents the cumulative total number of times the user's micro-gesture started on the cell of the smart device.

   b) Duration of touch (DT) of a cell, represents the duration of all the user's touch micro-gestures on a particular cell of a smart device touch screen.

   c) Pressure of touch (PT) of a cell, represents the pressure of all the user's touch micro-gestures on a particular cell of a smart device touch screen

   d) Frequency of usage (FU) of a cell, represents the cumulative total of particular cell accessed in a user's macro-gesture on the smart device touch screen.

   e) Last touch (LT) of a cell, represents the cumulative total number of times the user's micro-gesture ended on the cell of the smart device

2) Secondary features:

   a) Average touch pressure (APT) of a cell, represents the cumulative average touch pressure on a particular cell in a user's macro-gesture on the smart device touch screen. This feature is derived from the PT.

b) Average Duration of Touch (ADT) of a cell, represents the cumulative average touch duration on a particular cell in a user's macro-gesture on the smart device touch screen. This feature is derived from the DT.

c) Speed of user gestures (SG), represents the average speed of the user's micro-gestures in a particular macro-gesture on the smart device touch screen. This feature is derived from the DT.

4.3 Capturing User's Finger Gestures

In this section, we will present the algorithm used in our approach to generate the reward value R_{C_i} of each cell of touch screen of the smart device. The R_{C_i} represents the importance of cell and is estimated using the primary and secondary features of the touch screen of a smart device. Each cell of touch screen of the smart device is a state in the state graph of MDP [42, 43]. The MDP can be represented by a 4-tuple:

$$< \text{S, A, T, R}, \gamma >,$$

where S represents a finite set of cells on the user's smart device touch screen, A represents a finite set of actions, which represent the user's finger gestures, possible from every cell on touch screen of the smart device. T represents the set of probabilities $PR_{C_i}^{E_j}$ for the occurrence of every action E_j from each cell C_i. R represents the reward value R_{C_i} of C_i, which is derived from the features of the legitimate user's finger gestures on the smart device's touch screen as described in (5). The R_{C_i} is used as a metric for estimating the degree of importance of each cell on the user's smart device's touch screen. γ is the discount factor used for controlling the rate at which the MDP learns the user's finger gestures.

The collection of all the cells of a smart device's touch screen in a time period T_x, is denoted by S_{T_x}, can be represented as follows:

$$S_{T_x} = < C_1, C_2, \dots, C_i > \qquad (1)$$

where C_i is a cell of the touch screen of the smart device, where i = 1, 2, …, n, and *n* is the number of cells on the smart device's touch screen, and C_i is a 4-tuple

$$C_i = < FT_i, ADT_i, APT_i, FU_i, LT_i > \qquad (2)$$

where $FT_i, ADT_i, APT_i, and\ FU_i$ represent the features first touch, average touch duration, average touch pressure, and frequency of usage of C_i respectively.

$$D\left(C_i, CX_{T_x}\right) = \frac{ADT_i \times APT_i \times FU_i \times LT_i}{CX_{T_x}} \qquad (3)$$

$$\mathrm{S}\left(e, T_x, CX_{T_x}\right) = \frac{e \times CX_{T_x}}{T_x} \qquad (4)$$

The $D\left(C_i, CX_{T_x}\right)$ and $S\left(e, T_x, CX_{T_x}\right)$, given in (3) and (4), are used for measuring the importance of the cell and the importance of usage context to continuous user authentication, respectively. The CX_{T_x}will be defined and calculated using (6) in Section 4.4. The larger value of $D\left(C_i, CX_{T_x}\right)$ indicates that the specific C_i is more important in terms of user's touch pattern. The larger $S\left(e, T_x, CX_{T_x}\right)$ indicates that the user's finger gestures can provide more robust continuous authentication. $D\left(C_i, CX_{T_x}\right)$ and $S\left(e, T_x, CX_{T_x}\right)$ are then used for calculating R_{C_i}.

The R_{C_i} of the C_i of the smart device's screen are calculated as follows:

$$R_{C_i} = aD(C_i, CX_{T_x}) + bS(e, T_x, CX_{T_x}) \qquad (5)$$

where a, b and e are the weighting factors initialized during the application installation based on the type, the screen size, location, usage context, computational and space constraints of the smart device.

(5) is derived from our objective trade-off function (OTF) [44] for quantitatively measuring and incorporating performance and security in service-based systems. Since security configuration and traffic frequency vectors used in OTF are very similar to our primary and secondary features, we adapt the OTF to estimate the values of the cells based on the user's performance and security metrics requirement.

$PR_{C_i}^{E_j}$ is the probability metric for every edge E_j of C_i and is generated based on the user finger gesture movements on the smart device's touch screen. Here each edge represents transition between two states for a specific action. All the secondary features will be used for generating the update frequency of the probability and reward metrics in the state graph of the MDP because the secondary features are more stable and more accurately represent user's macro figure gestures. A major advantage of our approach is that it is flexible and designed to work even if a different technique is used to generate the R_{C_i} and $PR_{C_i}^{E_j}$ depending on the user's application requirements.

4.4 Usage context identification

In this section, we will present our technique to identify the usage contexts, which indicate how the user, in terms of finger gestures, is using the smart device over the time period, such as gaming, reading, and streaming videos. The usage context of the smart device is very important because it affects the nature of the user gestures. In order to have good accuracy for user authentication, it is essential to identify the smart device's usage context. On the other hand, the existence of different usage contexts implies that we need to change the MDP models to facilitate continuous authentication for the different usage contexts. Hence, it is desirable to have a small number of usage contexts for the user's smart device for quickly authenticating the legitimate user. From authentication point of view a larger number of usage contexts may not necessarily improve user authentication accuracy.

In our approach, we classify the user gestures into n usage contexts, where n is a preset number of usage contexts for a specific application. The classification will be done based on the primary and the secondary feature inputs. The idea here is to basically identify the usage context of the smart device usage. Each usage context has unique characteristics such as in gaming mode the user may use the touch screen of a smart device more frequently compared to other usage contexts. We try to use the primary and secondary features to discriminate the user gestures based on their unique characteristics and classify them. The equation for identifying the context is shown in (6),

$$CX_{T_x} = eNS(APT_i, ADT_i, SG_i) \qquad (6)$$

where CX_{T_x} is the usage context of the smart device's touch screen for a time period T_x, and APT_i, ADT_i, SG_i are the secondary features as mentioned in Section 4.2. The function NS is a normalized summation function which adds the normalized values of $APT_i, ADT_i\ and\ SG_i$. The major advantage of our approach is that it is flexible and designed to work even if a different technique is used to generate the CX_{T_x} based on the application requirement. The value of CX_{T_x} is then used to identify the smart device's usage context.

4.5 Continuous user authentication

In this section, we will discuss how to generate the plans [45] used for continuously authenticating the user of the smart device. Each plan is a sequence of states in a MDP state graph and represents a particular user's micro-gesture on the touch screen of the smart device. The continuous authentication technique for authenticating the legitimate user continuously is performed using the generated plans for all the micro-gestures. The cells on the touch screen of the smart device are used to capture the user's finger gestures. The MDP tuples described in Section 4.4 are specifically modeled to meet the requirements for our continuous authentication approach. The application provider initially sets the probability and reward metric of every edge and cell in the state graph respectively, based on the typical user finger gesture information. These parameters are then initialized on user's smart device during application installation or new user registration procedure. Based on the user's smart device usage, the parameters will be personalized to the individual user using the equations discussed in Section 4.4. Each user's usage context of the smart device has its respective probability and reward metrics.

In our approach, we use policy iteration technique [46] and value iteration technique [47], each of which is used to generate the plans for all the micro-gestures, because they are more efficient, accurate and relevant to perform continuous user authentication in a smart device. We have compared the performance of these two techniques using the illustrated example in Section 4.7. Other popular MDP techniques, such as modified policy iteration technique [48] are not suitable as they need more accurate and stable heuristics to use effectively, and generating these accurate and stable heuristics is very difficult because of the dynamic nature of the user finger gestures.

The following is the MDP policy generation algorithm:

```
1. Given a policy P
2. Loop:
        (a) Evaluate $V_P$ with (7)
        (b) For each $C_i$ in $S_{T_x}$ , set improved policy using (8)
        (c) Replace P with P’, where P’ is the new policy
   Until no improving action possible at any state
```

$$V_P(C_i) = R_{C_i} + \beta \sum_{C_{i'}} T(C_i, P(C_i, h), C_{i'}) \cdot V_P(C_{i'}) \tag{7}$$

$$P'(C_i) = \arg\max_{A} \sum_{C_{i'}} T(C_i, A, C_{i'}) \cdot V_P(C_{i'}) \tag{8}$$

where V_P is the value function of a policy P for a current cell i, R_{C_i} is the reward metric for the cell i, β is the discount factor which is used to control the importance of future reward, i’ represents a new cell neighboring to cell i

Policy generated by the MDP provides the optimal actions from each cell of the grid in probabilistic environment. Using the optimal policy and the first and last touch of a macro-gesture an optimal path is generated. The cumulative reward value for the optimal path and the cumulative reward value for the actual macro-gesture is compared to continuously authenticating the legitimate user. These optimal policies can be computed offline if needed

The value iteration technique [47] enables the MDP to constantly perform backward induction using the Bellman backup equation to generate the plans. In the value iteration technique, there is no initial plan and the plan is generated and then updated when the reward values of the cells are updated. The reward values for all the cells are initialized using (5) based on the degree of importance the typical user finger gestures, and is then adjusted continuously based on the legitimate user's finger gestures. The value iteration technique will generate plans that maximize the cumulative reward values of the plans in a probabilistic environment. The cumulative reward value of a plan is the summation of the reward value of all cells in a plan. The value iteration technique, after generating the plan, will keep updating the reward values of the cells till the plans stabilize. This is done to dynamically update the degree of importance of all the cells based on their neighboring cell reward values. This enables the system to be more accommodating to legitimate user's errors, thus reducing the false negative rates.

The policy iteration technique [48] works similarly to value iteration technique, but the difference is mainly that in policy iteration technique we initialize the technique with pre-selected plans and then use the backward induction to update the plan. The output of the policy iteration technique is a set of plans along with their respective cumulative reward values. The major advantage of the policy iteration technique with respect to the user continuous authentication is that unlike the value iteration technique, the policy iteration technique has prior knowledge on the plans, and hence converges quickly. Policy iteration technique is more likely to achieve plan convergence faster than the value iteration because the policy iteration technique iterates over the plans instead of the individual reward values of state as in the case of the value iteration technique. But, if the initial plan assignment is random or incorrect, then the policy iteration technique may take longer to converge or generate suboptimal plans, either of which will affect the continuous user authentication.

In summary, each of the value iteration technique and the policy iteration technique will generate a set of plans along with their cumulative reward values. These plans will then be ranked based on their cumulative reward value, importance to the observed user's micro-gestures and the smart device usage context. The cumulative reward values of the ranked plans are then used as benchmark data for continuously authenticating the user's macro-gestures on the touch screen of the smart device. The dynamic nature of the MDP enables the reward metrics and the probabilities to keep changing continuously along with the user's evolving gestures. This is unlike the supervised learning techniques, where the learned pattern does not change with the evolving user finger gesture.

The applicability of each of these two techniques depends on the user's application requirements. In some cases, one technique is better than the other and vice versa. Since the policy iteration technique has the prior typical user finger gesture information, the technique can perform continuous user authentication faster, but with relatively acceptable error rates which may be suitable for normal user authentication. However in highly critical applications, more robust continuous user authentication is needed. In such applications, the value iteration technique will be more suitable because it has no biases on user finger gestures and hence can achieve more accurate continuous user authentication.

4.6 Evaluation

In order to evaluate our approach we collected the actual user usage data and partly used the dataset used in [41]. The unified dataset consists a total of 20 graduate students at ASU in the age group of 21 to 29. There were no restrictions placed on the users, they were instructed to use their smart phones as they would do so normally. The users can browse web pages, including news, online forums, social network websites, etc., or use the installed apps, such as twitter, facebook, browsers, etc. Users were not required to continuously use the smartphone.

The evaluation of our approach was done for a grid size of 8X6, which provides us 48 cells. The purpose for choosing the 8X6 grid is to provide us adequate fidelity of finger gesture information. Additional constraint on the grid size is also computation overhead, as larger grid size has higher overhead. The 8X6 grid is shown in Figure 3. For the purposes of visualization we constructed Figure 3, which shows the distribution of the reward values across the various cells of the grid for a sample legitimate user. The scale for Figure 3 is as follows, colors in the green spectrum indicate high cell usage and the colors in the red spectrum range indicate low cell usage. The magnitude of the usage is illustrated by the brightness of the colors, darker colors representing higher magnitude and lighter colors representing the lower magnitude. As shown in the Figure3, each of the cell in the grid is named as Cx, where x ranges from 1 to 48.

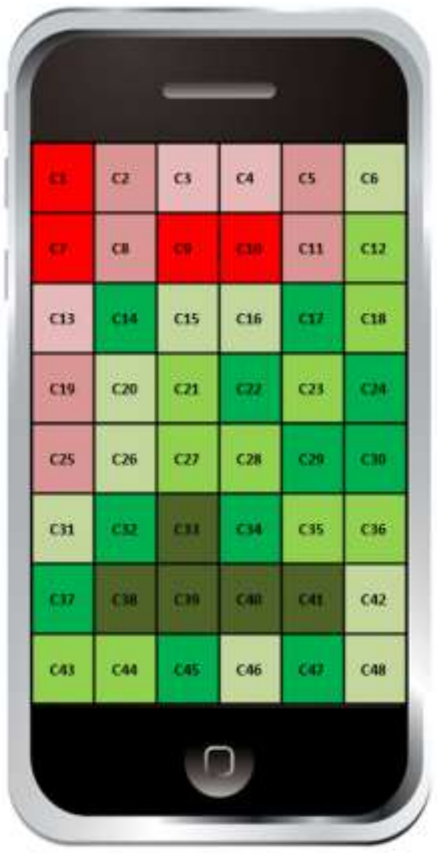


Figure 3: Sample Finger Gesture Usage Pattern of Legitimate User

Figure 3 was generated from a selected user usage data of the touch based smart phone. We can infer from the diagram that the user is most likely a right hand user as there is a heavy usage of the bottom left corner of the touch screen which is typical pattern observed in the right hand users. The color and brightness distribution also shows a significant vertical and horizontal scroll gestures, which is consistent with the general usage pattern of the smart phone where scroll gestures contributing significantly larger proportion of the finger gestures [41].

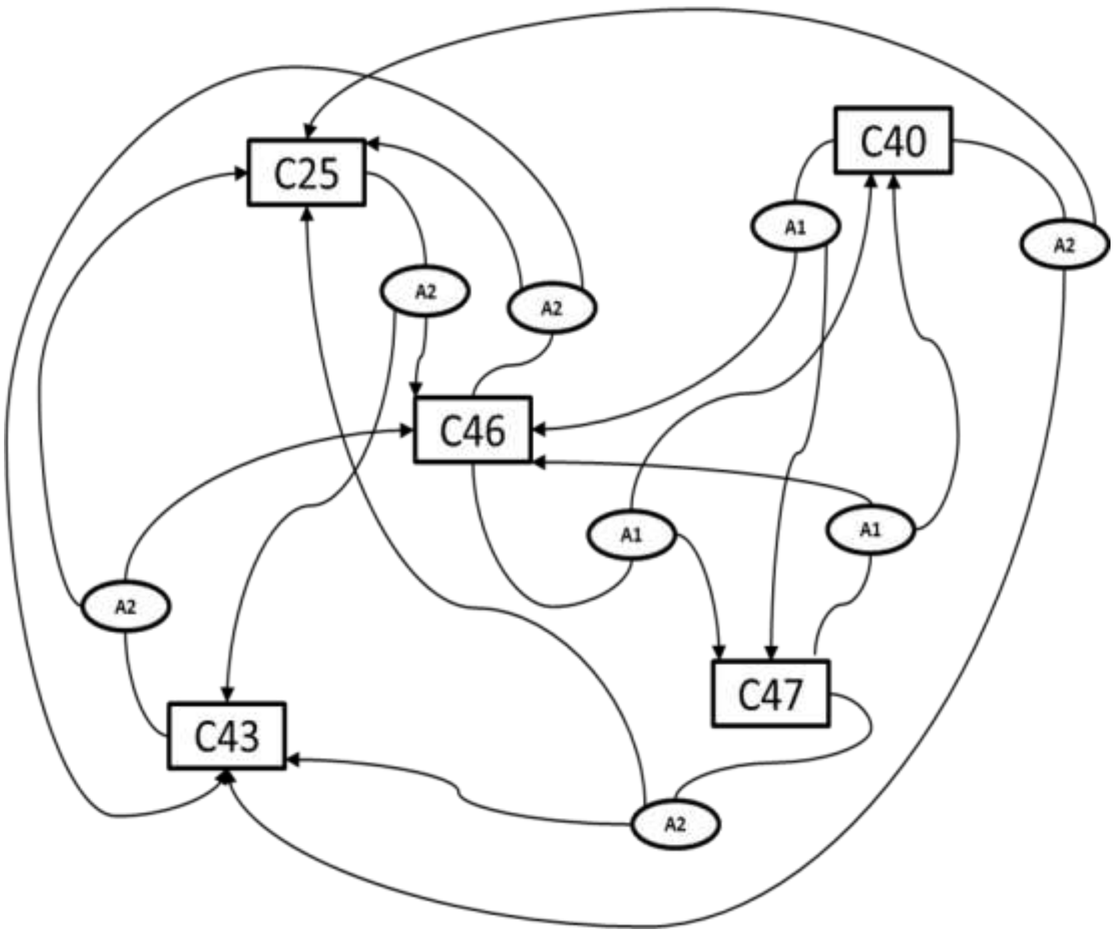


Figure 4: Sample State Graph for Finger Gesture Usage Pattern of Legitimate User

Figure 4 illustrates the actions and its effect on the state transitions. As mentioned in our overall approach in Section 4.1, we consider two kinds of gestures micro-gestures which is relates to action A1 and macro-gestures which is relates to action A2. Each of the rectangular box shown in Figure 4 is a specific cell on the touch screen of the smart device. We can observe from Figure 4 that actions in A1 category resulting in neighborhood state transitions, meaning the next state will be a neighboring state to the current state. Actions that relate to A2 result in state transitions to non-neighboring states. Note that each state transition is probabilistic in nature and hence has a probability value assigned to it. For example the transitions among the C46, C47, C40 and C46 with only A1 actions represents the user is exhibiting a uninterrupted touch gesture on the touch screen of the smart device. Similarly, transitions among the C25, C43, C46 and C43 with only A2 actions represents the user is exhibiting a skip touch gesture on the touch screen of the smart device.

The application was designed to monitor and record the users' finger movements on the touch screen of the smart device. For this experiment we fixed the number of states for MDP to be 48, to make it robust and to ensure cross-device functionality. The application automatically populates the MDP probability, reward and transition values required for the user authentication based on the observed user finger movement data. For the experimental purpose, the application was designed to passively authenticate the users, meaning that the results of this authentication process were not shown to the test users but were stored for later analysis. The data from all the devices were retrieved and the initial analyses have shown promising results.

Our overall evaluation followed the below steps of our approach is shown in Figure 3,

Step 1) Continuously collect the sensory data from the selected users' smart device during their using of the smart device. Follow the technique presented in [41] to extract the data from the mobile device.

Step 2) Analyze the sensory data flows collected during the user's usage session of the smart device in Step 1 to extract the necessary information to construct the required features. The output of this step is the features which are used for grid value generation (GVG) process.

Step 3) Using the features constructed in Step 2 to run the GVG algorithm on user's smart device for continuously updating the MDP state graph during the user's smart device usage session. The inputs for the GVG algorithm include the attributes of user gestures on the smart device screen, such as pressure, speed and usage frequency.

Step 4) Identify the context in which the touch screen of the smart device is being used by the legitimate user using the primary and secondary features.

Step 5) Generate the plans which are the sequences of cells/states in the MDP state graph representing the user's gestures, along with the plans' respective cumulative reward value. These plans are generated using our technique that uses the MDP algorithm.

Step 6) The cumulative reward values of the plans are then used as benchmark data to continuously authenticate the legitimate user of the smart device based on the cumulative reward

As mentioned earlier each of the legitimate user usage data was collected for a week. The usage data of the individual legitimate user was then used to personalize the MDP state graph to reflect their respective usage pattern. Each of the 20 participant data was used in our evaluation. In order to make our evaluation realistic we generated the negative list of the figure gesture data of potential malicious user by randomly selecting the finger gesture data from non-legitimate users. The process of random selection was used in order to reduce the selection biases during the negative list generation process.

The error rates for the various observation windows for different contexts are shown in Figures 5 and 6. Figure 5 shows the False Acceptance Rate (FAR) spread across observation windows. The FAR metric measures the acceptance of the non-legitimate user as a legitimate user by the authentication algorithm. The FAR is considered as one of the most important metric as it measures the security performance. The average FAR for our continuous authentication algorithm ranges from 8%, for shorter observation window, to as high as 24%, for longer observation windows.

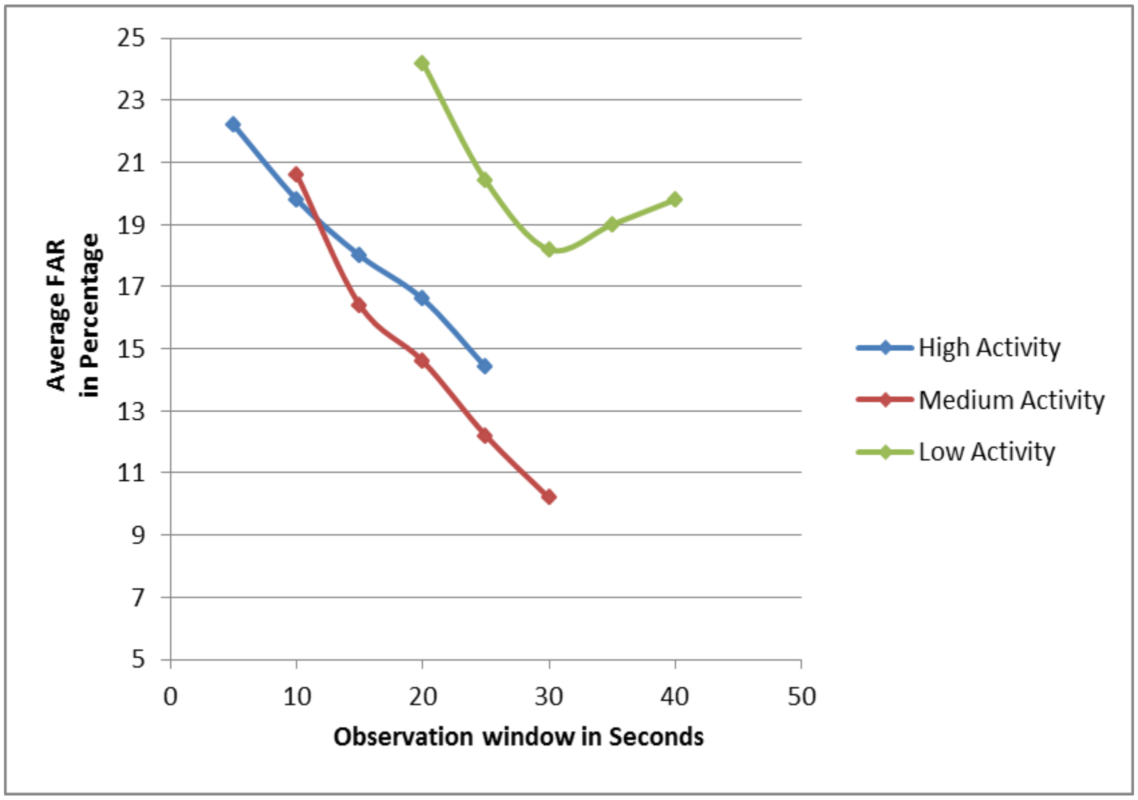


Figure 5: Spread of Average False Acceptance Rates of Our Continuous Authentication Approach for Various Contexts

As we can observe from Figure 5, the medium activity context has comparatively lower FAR than the high activity and the low activity contexts. Intuitively we should expect lower FAR for high activity context than the medium activity context, since input figure gesture frequency is higher. Generally more information means lower FAR, but we did not observe this except for one observation window of 10. We suspect that that the higher FAR of high activity context is due to high finger gesture information noise. The continuous authentication algorithm may be unable to accurately discriminate against non-legitimate users in low activity context due to the fact that there is insufficient finger gesture information.

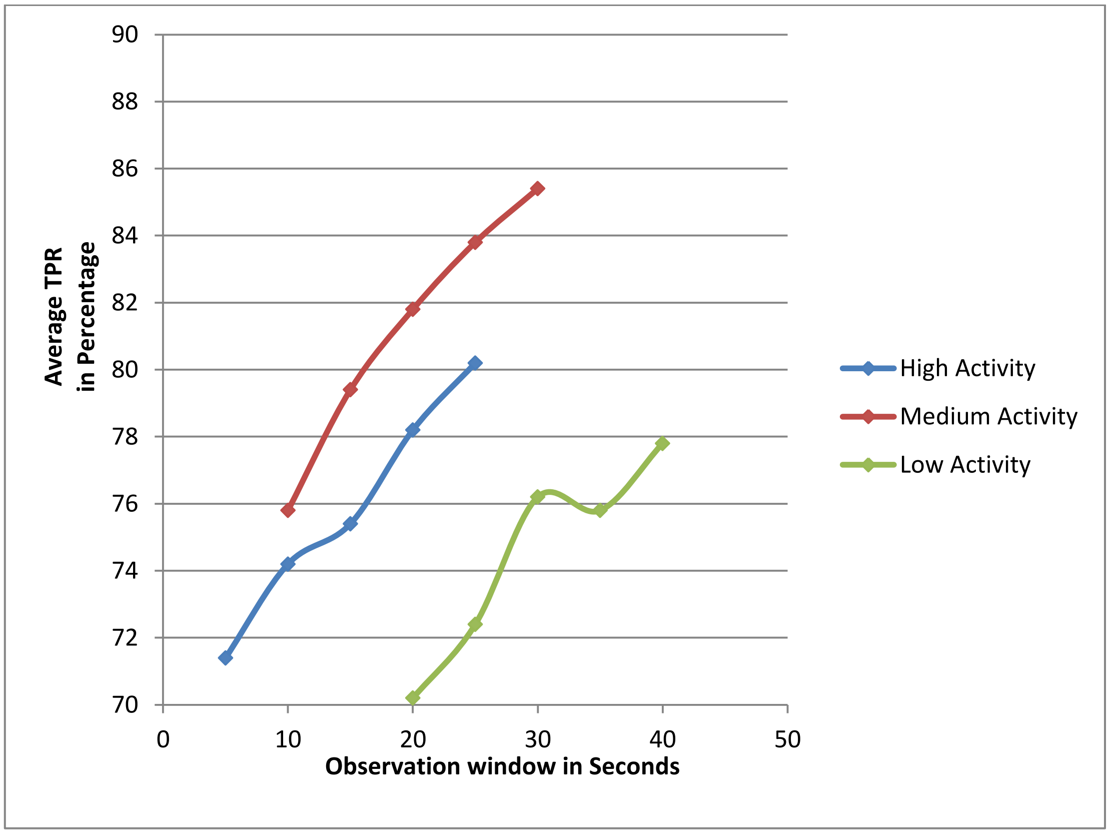


Figure 6: Spread of Average True Positive Rates of Our Continuous Authentication Approach for Various Contexts

As we can observe from Figure 6, the medium activity context has comparatively higher TPR than the high activity and the low activity contexts. Similar to the FAR figure, intuitively we should expect higher TPR for high activity context than the medium activity context, since input figure gesture frequency is higher. But, as in the case with FAR figure, we did not observe this for any of the observation windows. We suspect that that the lower TPR of high activity context is due to high finger gesture information noise. The continuous authentication algorithm may be unable to accurately discriminate against non-legitimate users in low activity context due to the fact that there is insufficient finger gesture information.

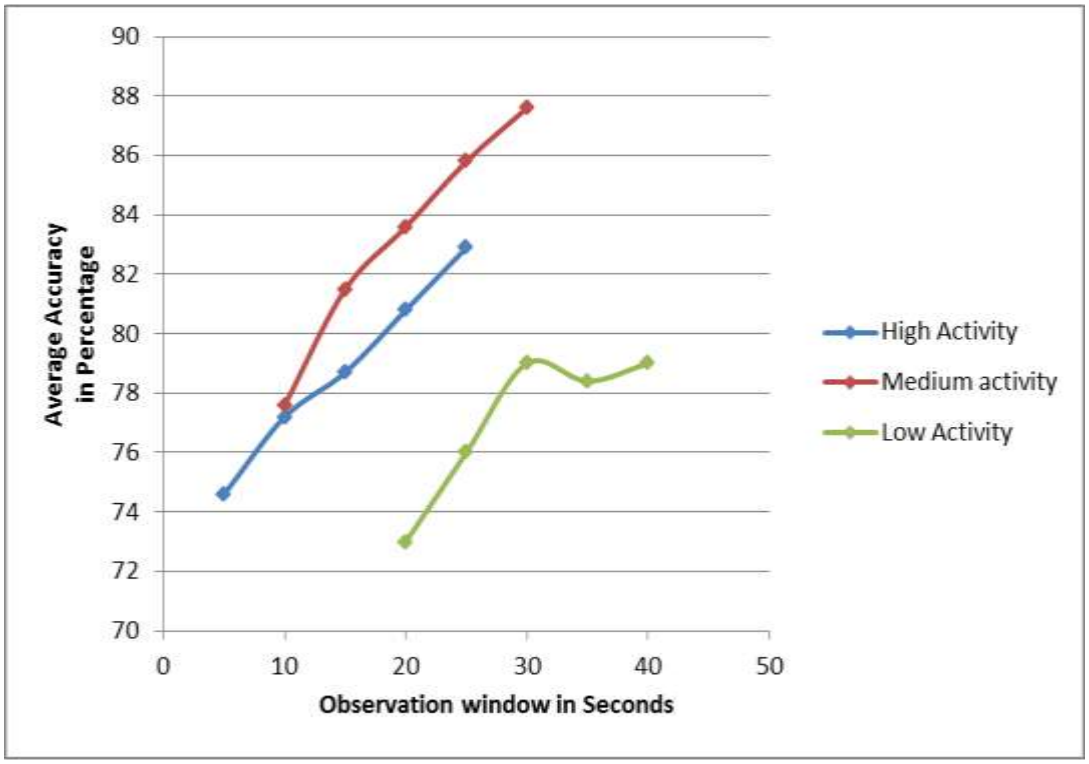


Figure 7: Spread of Average Accuracy of Our Continuous Authentication Approach for Various Contexts

Figure 7 shows the average accuracy rate of our continuous authentication algorithm. The observations among the high activity, medium activity and the low activity levels are very similar to the TPR graph as expected. Based on the observed FAR, TPR and average accuracy metrics we conclude that our approach to continuously authenticate legitimate user performs relatively well. We also recommend the observation windows for the high activity, medium activity and low activity levels to be 10, 20 and 30 seconds respectively. In order to effectively secure the touch based smart device during the observation windows we recommend restricting the access to the critical applications of the smart device till the user is authenticated.

In addition we benchmarked our continuous authentication approach with the existing techniques to both authenticate and re-authenticate the legitimate user of the smart device. The major categories of the benchmarks that we chose were usability, deployability and security. The rationale behind choosing these categories was that they were some of the most important metrics for evaluation of authentication approaches. The sub-categories for each of the major categories, as shown in Table 1, were chosen to effectively discriminate and compare the various authentication and re-authentication techniques.

Table 1: Comparative Evaluation of Various User Authentication Schemes

			Usability						Deployability				Security							
Category	Scheme	Reference	Scalable-for-Users	Memorywise-Effortless	Nothing-to-Carry	Infrequent-Errors	Physically-Effortless	Efficient-to-Use	Ease-of-Deployability	Ease-of-Personalization	Ease-of-Adaptation	Robustness	Resilient-to-Targeted-impersonation	Resilient-to-Throttled-Guessing	Resilient-to-Physical-Observation	Resilient-to-other-Verifiers-Leaks	Resilient-to-Phishing	No-Trusted-Third-Party	Resilient-to-Theft	Conscious Continuous Authentication
(Incumbent)	Digit/Pattern Password	[8]	●	○	●	●	●	●	●	●		●				●		●		
Biometric	FingerPrint	[8]	●	🟢	🟢	⭕	●	○	⭕	●	○	○	🟢	🟢	🟢	○	🟢	○	○	
	Iris	[21]	●	🟢	🟢	⭕	●	○	⭕	●	○	○	🟢	🟢	🟢	○	🟢	○	○	
	Voice	[21]	●	🟢	🟢	⭕	⭕	⭕	⭕	●	○	○	◎	🟢	🟢	○	🟢	○	○	
	Camera	[22]	●	🟢	🟢	⭕	⭕	⭕	⭕	●	○	○	◎	🟢	🟢	○	🟢	○	○	
	Gyroscope	[23]	●	🟢	🟢	⭕	⭕	⭕	⭕	●	○	○	◎	🟢	🟢	○	🟢	○	○	
Cognitive	Unob-Auth	[20]	○	🟢	🟢	○	○	⭕	⭕	🟢	○	○	◎	🟢	🟢	○	🟢	○	🟢	○
	Rhy-Auth	[62]	○	🟢	🟢	○	○	○	⭕	🟢	○	○	◎	🟢	🟢	○	🟢	○	🟢	
	Multi-Touch	[18]	○	🟢	🟢	○	○	○	⭕	🟢	○	○	◎	🟢	🟢	○	🟢	○	🟢	
Adaptive Cognitive	Contin-Auth	[12]	🟢	🟢	🟢	○	🟢	○	🟢	🟢	🟢	○	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢

●= offers the benefit, ○= almost offers the benefit, no circle = does not offer benefit
Green outline = better than incumbent passwords, Red outline = worse than incumbent passwords

Note that both the accuracy, FAR and TPR were based on the user usage data for one week. Our algorithm is reinforcement based, hence it tends to get better with higher usage. We expect the FAR and TPR metrics to improve significantly as the algorithm gets used continuously by the legitimate user for longer periods of time.

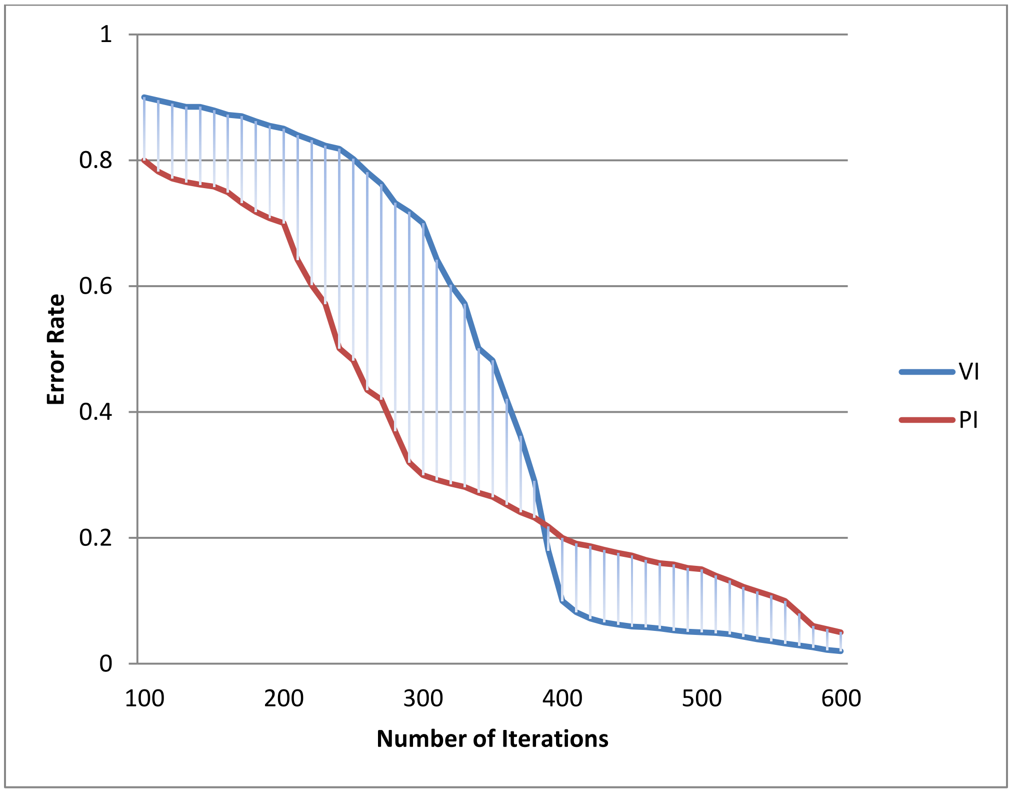


Figure 8: Comparison of Error Rates for Value Iteration and Policy Iteration Techniques of MDP

Additionally we did a comparative study between two variations of MDP implementation which yielded graph as shown in Figure 8. Policy iteration (PI) technique, shown as red line in Figure 7, performs faster continuous user authentication and has higher error rates since it is biased with the prior typical user finger gesture information. Value iteration (VI), shown as blue line in Figure 7, is more robust and hence is applicable for highly critical applications and has a slower convergence rate. It also has no biases on user finger gestures and hence can achieve more accurate continuous user authentication.

4.7 Further Improvements

In this chapter, we have presented an approach that uses MDP and touch gestures of the user to continuously authenticate the user on a touch-screen smart device. This approach incorporates continuously learning, adaptation and validation of the user’s finger gestures, and also estimation the usage context of the smart device. The major advantage of this approach is that it does not require the user to consciously provide inputs for training purposes and does the learning, adaptation and validation of the user’s finger gestures in the background, invisible to the user. By continuously learning the user gestures, our system will be able to capture important unique user characteristics and also adapt to evolving user’s finger gestures. This enables our system to build an accurate adaptive user gesture pattern that is very effective and efficient for continuous user authentication. Another major advantage of our approach is to have better authentication accuracy by treating uninterrupted user finger gestures over a short time interval as a single gesture for continuous user authentication

We need to conduct experiments to determine suitable weighting factor values for our approach. In addition, we plan to further reduce the memory and computational resources required for our approach by factorizing the cell attributes in MDP. We plan to integrate this approach with the development of smart IoT environment [13, 49]. The size of the grid and the shape of the individual cell also play a vital role in determining the authentication accuracies, and hence we plan to conduct further research in this direction. We also plan to extend this approach to include the use of user-specific information, such as device type, user age, gender, geographic location, operating system to better incorporate user’s characteristics.

Chapter 5

INTELLIGENT ADAPTATION OF DEVICE STATES IN IoT ENVIRONMENT

U-things (smart devices) are being used everywhere now-a-days, more and more smart devices, such as smart phones, central air conditioners, TVs, and smart ovens, are coming into workplaces and homes. Recent surveys [1, 2] have shown that this trend will increase rapidly in the future. U-things are physical things with attached, embedded or blended computers, networks, and/or some other devices, such as sensors, actors, and e-tags [50]. Although most of these devices are designed to interact with each other, they are mostly used with few interactions with their surroundings. In order to fully realize the potential of u-things to satisfy a broad range of requirements of many users efficiently, such as satisfying various QoS as well as functional requirements, it is necessary to have intelligent mobile IoT applications incorporating interactions of u-things as well as non-u-things with other smart devices in their environments. A non-u-thing is a physical thing without network communication interface to interact with its environment, such as food heaters, monitors.

Current approaches to developing intelligent mobile IoT applications use only many-to-many communication among u-things with minimal automation, which often results in large communication overhead and may require complicated handshake protocols. In this chapter, we will present an intelligent planning approach to developing cloud-based mobile IoT applications with reduced communication overhead by using only one-to-one communication between a u-thing and one of the control modules of its application environment. We consider a mobile IoT application as a physical space consisting of u-things, non-u-things and users, in which these devices and users operate. Developing an intelligent mobile IoT application includes enabling the interactions of the u-things, non-u- things and users with their surroundings.

In the past, the number of the u-things in a users' environment has been relatively small, and they were easy to operate. In such an environment, users can set and/or select devices' modes of operation. But, now the number of u-things in user's applications is rapidly increasing, and many u-things have more modes of operation. Users are usually not expected to set and/or select their smart devices' modes of operation due to the increasing complexity involved in the causal relationship the devices has on their environments. However, each device's mode of operation has to be set so that the goals of the intelligent mobile IoT application, such as maximizing performance, security, and energy efficiency, can be achieved. It is important to include both u-things and non-u-things in our approach because non-u-things may be needed to achieve the users' goals.
The main contribution of this chapter is to present an approach to intelligently planning the device actions required to achieve the users' mobile IoT application requirements efficiently. Our approach also includes a technique for service providers to assess the states of all the devices in IoT applications. The intelligent planning will be performed based on the dynamic assessment in a mobile IoT application environment, which is accomplished by using learning and planning algorithms through mobile cloud technologies. A plan is nothing but a sequence of device actions to be executed by the devices/things in the mobile IoT application.

In this chapter, Section 5.1 contains our overall approach. In Section 5.2, we will discuss how the service providers for an intelligent mobile IoT application will analyze device states. In Section 5.3, we will discuss our approach to automatically label the IoT devise states. In Section 5.4, we will discuss our approach to dynamic assessment of the performance of the devices. In Section 5.5, we will discuss our planning technique for developing an intelligent mobile IoT application. In Section 5.6, we will present an example to illustrate our approach, and in Section 5.7 we will present some simulation results to show the effectiveness of our approach. In Section 5.8, we will discuss conclusions and future work.

5.1 Approach

The research tasks proposed in this section includes developing approach that uses regularized least square and Markov Decision Process for intelligent planning of device actions in IoT application environments. The specific goals of our approach are: (1) assist the user(s) of the IoT application in securing their environments, (2) observe and monitor user(s) behavior to plan device actions that complement user behavior and preferences, and (3) ensure that the user requirements are satisfied with minimal resources. Each of the above items will be discussed in detail in the Section 5.1.

In our approach, we consider three types of entities in an intelligent IoT application environment: users, manufacturers and smart environment service providers SESP. The primary functions of these entities include the following, (1) The users provide the SESP with the functional and quality requirements, and application environment, (2) The manufacturers provide the state information and device specification manuals of the u-things and non u-things requested by the SESP, and (3) The SESP will analyze the above requirements, obtain the list of u-things and non u-things, and develop machine learning algorithm to assess the devices in the application environment.

We define intelligent environment tiers IET to indicate how well a device's state conforms to the users' specified application, such as satisfactory, marginal, and unsatisfactory. The number of IET required for a IoT application will be determined by the SESP after analyzing the users' application specification. An intelligent IoT application will consist of u-things, non u-things and users, whose actions and/or performance collectively will satisfy the users' IoT application requirements. In our approach, we will use a low-cost device authentication mechanism [3] to authenticate all the u-things. We will use our prior results to ensure the trust management and confidentiality protection requirements in cloud environment are efficiently handled [49, 51, 52]. Data reported from those u-things which fail the authentication will not be accepted.

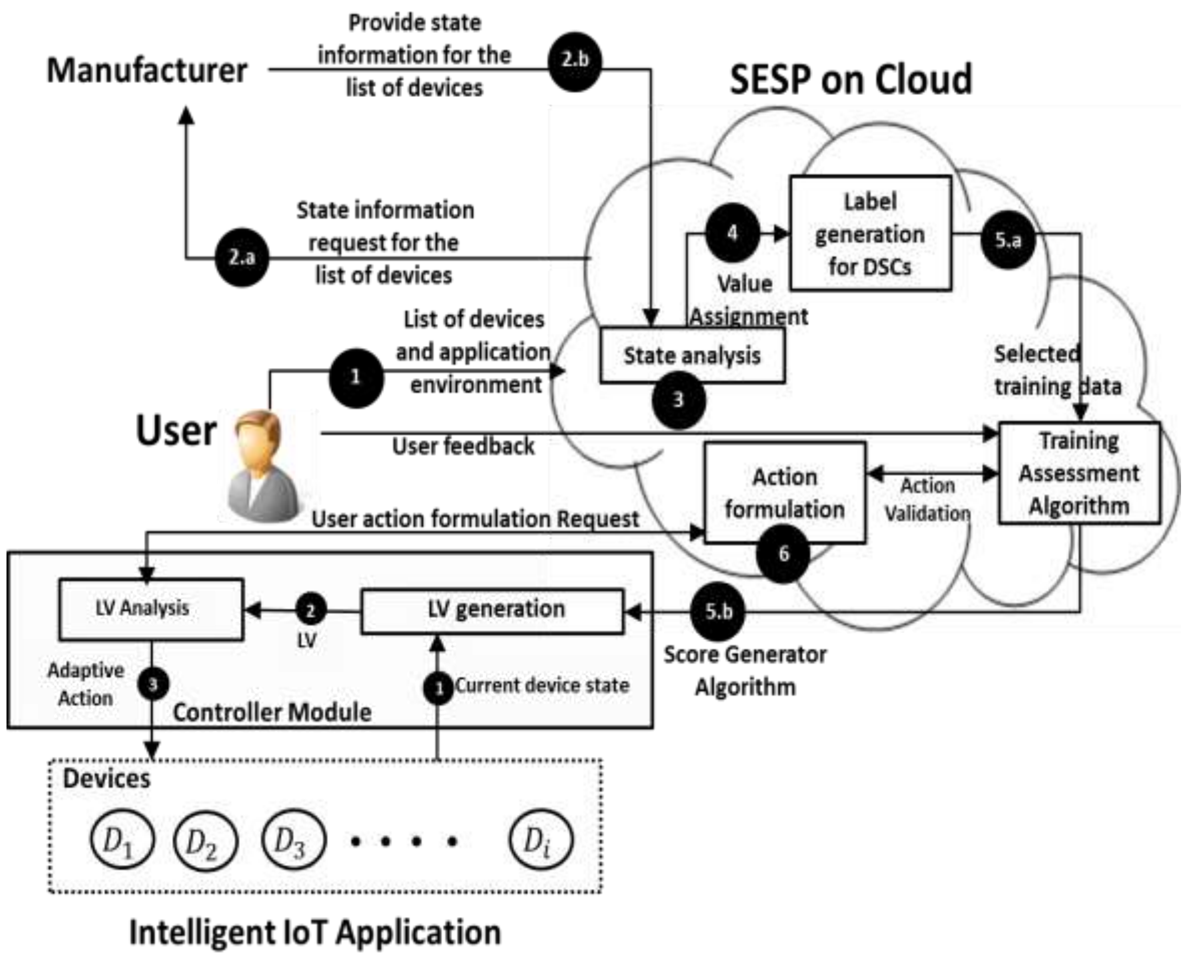

Manufacturer
Provide state information for the list of devices
2.b
SESP on Cloud
State information request for the list of devices
2.a
4
Label generation for DSCs
5.a
Value Assignment
List of devices and application environment
State analysis
1
3
Selected training data
User
User feedback
Training Assessment Algorithm
Action formulation
Action Validation
User action formulation Request
6
LV Analysis
2
LV
LV generation
5.b
Score Generator Algorithm
Adaptive Action
3
Controller Module
1
Current device state
Devices
D_1
D_2
D_3
D_i
Intelligent IoT Application

Figure 9: System Diagram for Our Approach to Intelligent Adaptation of Device States in IoT Environment

Our overall approach [13] is depicted in Figure 9, including various entities and their interactions. Our approach involves the following steps,

Steps on SESP side:

Step 1) A user provides the functional and quality factors that are required by SESP for enhancing the security of the IoT application environment.

Step 2) The SESP identifies the list of u-things and non u-things, and their state information from the registry of Things, which is initially generated with the information provided by the manufacturers of u-things and non u-things.

Step 3) The SESP analyzes the state information of each u-thing and non u-thing based on the ranking of importance of all the states with respect to the users' IoT application specifications to assign value to the state of the device.

The SESP also assigns an impact index IM for u-things and non u-things, which reflects the impact the devices, has on certain quality factors of the IoT application.

Step 4) The SESP uses ALA our technique to automatically assign the IET labels for selected Domain State DSs. More details on our ALA algorithm will be presented in Section 5.3. DSs are the combinations of observed u-things and non u-things states at any particular time in the IoT application environment.

Step 5) The SESP uses the labeled DSs in Step 4) as the training data to generate the trained machine learning algorithm using supervised learning techniques.

Step 6) SESP uses our technique that uses markov decision process MDP planner on its cloud servers for action formulation and the resultant domain state obtained from executing formulated actions of application domain is validated by the trained machine learning algorithm to check for plan's effectiveness with respect to the user requirements.

Steps on user intelligent application:

Step 7) The SESP provides/installs a controller device (CD), which includes trained machine learning algorithm in users' IoT application to monitor all u-things states. The CD can be installed in a smart phone/tablet which is designed to access cloud through the network.

Step 8) The trained machine learning algorithm uses all the u-things' states and the given state of non u-things of a particular time instant to categorize the DS. The states are collected by the CD using communication systems such as bluetooth, zigbee, wi-fi.

Step 9) CM then transmits the data to cloud servers where Step 6) is performed to formulate the device actions to adjust the necessary u-things action/operations in real time to reach the users' expected goal through CD. Since CD cannot control non u-things, the CD only provides recommendation on the required actions of non u-things to the users.

5.2 Dynamic IoT State Assessment

In this section we will primarily discuss the development and deployment of the learning algorithms, namely RLS (regularized least squares) and ALA (Automatic Labeling Algorithm), for the dynamic assessment of the IoT environment. The *SESP* will primarily use the ALA to automatically label the training dataset with minimal utilization of the domain experts, and use RLS algorithm for the dynamic IoT assessment. The rationale for developing two algorithms, RLS and ALA, for IoT environment assessment is for accommodating the real-world problem of availability/feasibility of extensively using domain experts. Any restriction on the availability of domain experts will have major impact on labeling the training data. In such cases, we propose the ALA algorithm which is an unsupervised learning algorithm. The RLS algorithm is a supervised learning algorithm hence it requires labeled training dataset. The ALA is designed to organize the dataset based on the similarity of their individual data points to one another. A few of the data points are randomly selected and manually labeled by the domain experts and the rest of the data points are automatically labeled by the ALA. The details of the ALA and RLS algorithms will be discussed in Section 5.3 and 5.4 respectively.

The primary responsibility of the *SESP* is to assign device values for each device and train the machine learning algorithm, which will be discussed in Section 5.3. The *SESP* obtains the state information and device specifications manual from manufacturers. Attribute of device state is defined as a feature of a device which affects the mobile IoT application, and can be controlled by a user or control units. Examples for attributes of an air conditioner are blower, temperature, and time. *SESP* performs detailed analysis on the state information and attributes of the state, if needed the *SESP* may perform device testing to gather required information for state analysis. After the *SESP* gathers the necessary information, it performs state importance analysis using the domain experts, where it looks at the ranking of importance of the device states with respect to the mobile IoT application and assigns *UV*. The state of the mobile IoT application, called the domain state *DS*, consists of the total states of the individual devices. The *DS* at time t, DS_t, is represented as follows:

$$DS_t = < D_1, D_2, \dots, D_i, \dots > \qquad (1)$$

where D_i is the total state of the device i, i = 1, 2, …, which is a 4-tuple

$$D_i = < N_i, L_i, S_i^t, C_i, E_n >, \qquad (2)$$

where N_i, L_i, S_i^t and C_i represent the name, location, state and context of device i, respectively. The *UV* and *IM* of the devices in a mobile IoT application are calculated by the SESP using our objective tradeoff function [44] as shown in (3) and (4). E_n represent the possible events that can occur from current state S_i^t. E_n contains state changes such as on, off, sleep

$$UV_{S_i^t} = F(N_i, S_i^t, AD) \qquad (3)$$

where $UV_{S_i^t}$ is the value of the state of the device i at time *t*, and *AD* is the application domain preference set by the user.

$$IM_i = F(N_i, L_i, C_i) \qquad (4)$$

where IM_i is the impact factor for each of device i.

In the IoT environment, when the number of total states and the devices space are very large, the *SESP* can establish a subset of benchmark states and assess the similarity of the states of each device to the benchmark data, and then assign a *UV* for the state of a device [53, 54]. This method has little overhead because the attributes of the state have to be defined in such a way that the similarity algorithm can compare the attributes of a devices' state with the benchmark data. The alternative to this method is ranking the importance of attributes of all states. The *SESP* then can select a set of highest and/or lowest ranked attributes, whose occurrence will trigger a large change in the *UV* assignment of a state of a device.

$$LV_{DS_t} = \frac{1}{\mu N_{DS_t}} \left(\sum (\alpha IM_i * \beta UV_{S_i^t}) \right) \qquad (5)$$

where LV_{DS_t} is the label value for each domain state at time t, $N_{DS_t}$is the number of the devices in a given domain state at t, and μ, β and α are the variables that can be used to assign weight for each parameter based on users' application specification.

The SESP generates the *DS* that are representative of the users' application requirements. A small subset *DS'* is randomly selected from the overall DS. Equation (5) is used for calculating the LV_{DS_t} for each data instance in *DS'*. Note that the equation (5) is only used to calculate *LV* of *DS'* to be used as a part of the training data. Each of the *DS'* is assigned to one of the IET classes by the domain experts based on their LVs generated using equation (5).

5.3 Automated Labelling

The primary objective of the ALA learning algorithm will be to minimize the use of domain expert inputs and at the same time provide relatively accurate assessment on the IoT environment. One of ways of minimizing the utilization of the domain experts is to minimize the practice of manual labelling, which is one of the most time consuming process. Towards achieving this objective we propose a technique to exploit the property of similarity to organize the whole training dataset and heuristically assign labels to unlabeled data instances. The ALA is derived from our prior work [55]. We will present more details in the below subsections.

The ALA will use the modified k-means clustering technique with Mahalanobis distance as the distance metric to measure the similarity among the data instances in the dataset. The primary rationale for using Mahalanobis distance metric is to accommodate the variance within the cluster. The traditional implementation of the k-means clustering technique with Euclidean has major limitations. One of the major limitations is that it assumes every cluster to be spherical in shape. When the data instances distribution of the dataset does not conform to the spherical cluster shape then the accuracy of the clustering reduces significantly. This problem is especially acute in the IoT environments which are dynamic in nature. In order to accommodate the non-spherical cluster shapes and variances even within the classes we will use the Mahalanobis distance metric for clustering purpose. The covariance matrix required for generating the initial shape of the cluster will be calculated from the data instances in the *DS'*.

The individual devices states' composing the DSs form the feature set and the number of IET classes correspond to the number of the clusters. The k-means clustering algorithm is an unsupervised learning algorithm hence it does not need labeled dataset. Since the ALA is required to assign IET classes to each of the data instance in the overall DS, the clusters generated by the modified k-means algorithm needs to be correspond to the IET classes. In order to associate the cluster to the IET class, we follow the "first-past-the-post voting" principle. A cluster may be composed of data instances having multiple IET labels, and a cluster with highest count of IET labels will be assigned the respective IET class.

The following steps illustrate the ALA,

Step 5.3.1) Generate a subset *DS'* from the overall *DS* through a process of random selection.

Step 5.3.2) Calculate the covariance matrix using the data instances in DS'.

Step 5.3.3) Use the modified K-means algorithm to generate and update the clusters by organizing the data instances in *DS*.

Step 5.3.4) Use the voting technique to assign the IET class to the generated clusters with the IET classes assigned in *DS'*

Step 5.3.5) Assign the IET label corresponding to the cluster to all the data instances within the cluster.

Step 5.3.6) Repeat Steps 5.3.3) to 5.3.5) until an accurate ALA is generated for automated IET label assignment.

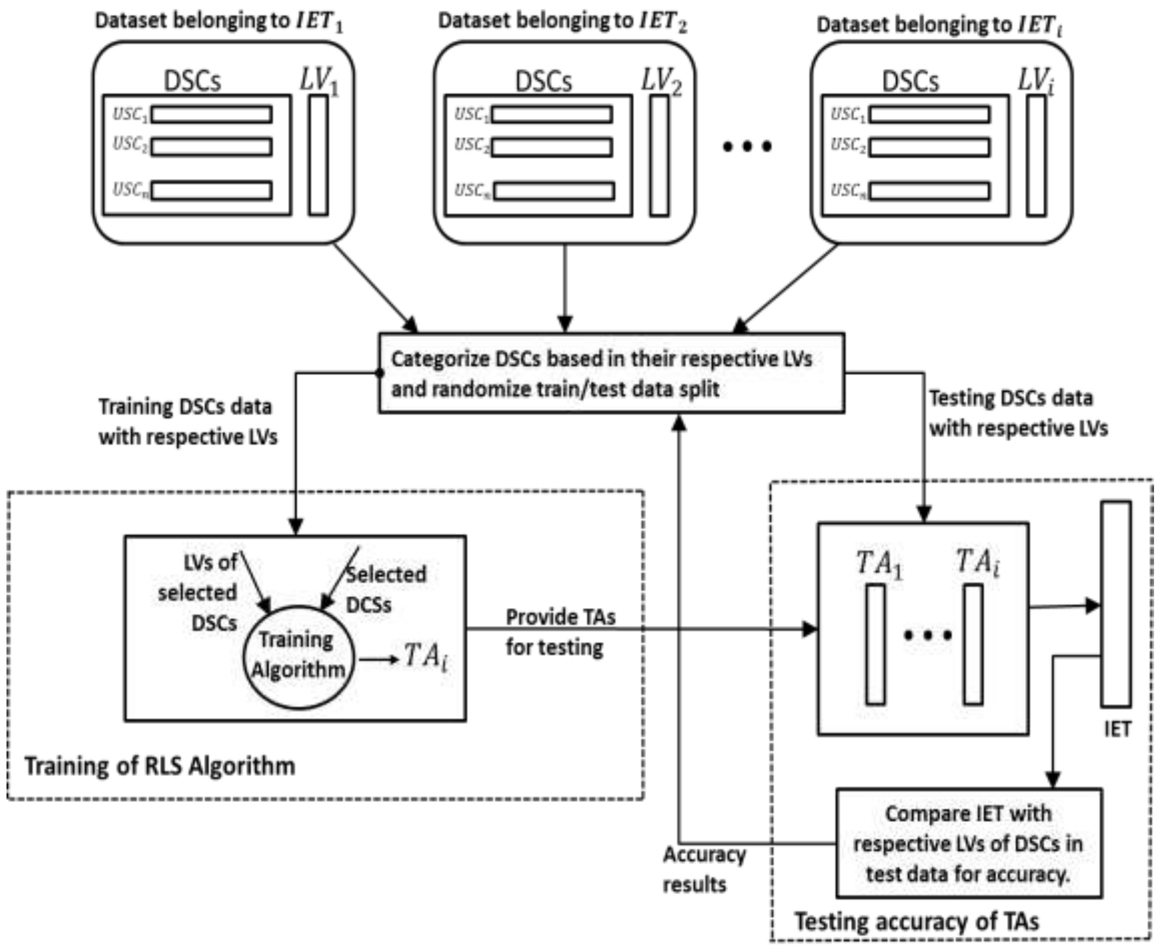
The accuracy of the ALA depends on both the labelling accuracy *DS'* which is performed by the domain experts, and the generation of the overall DS dataset which needs to be representative of the users' application requirements. The *SESP* uses the input from the *CM* in the user's mobile IoT application to maintain the logs of *DS* occurrences in the environment. The *SESP* uses these logs to generate the probability metrics for the occurrences of the devices' states in the user's mobile IoT application. Then the *SESP* can generate a list of frequently occurring states and a subset of *DSs* for the training data. Then the *SESP* can ensure that the mobile IoT application is adequately represented in the training data that is generated. The labelled data instances of the overall *DS* dataset will be used by the RLS algorithm to generate the model for assessing the dynamic IoT environment.

5.4 RLS Learning Algorithm

The objective function in the least square regression algorithm [56] is generally prone to overfitting and the dynamic nature of the IoT environment will likely make the overfitting problem much worse. This is due to the fact that dynamic environments generally require priors to generate accurate models, and hence in our approach we penalize the least square algorithm by adding a regularization variable. The regularization of the least square objective function can be either the l_1 or the l_2 norm depending on the users' requirements on the adaptiveness of the learning algorithm.

As mentioned in the above section, the ALA automatically labels the individual data instances of the *DS* dataset. The *DS* dataset is now used as the training data for training the RLS algorithm. The preparation of the dataset will include k-fold cross-validation, where the k is 10. More on development and usage of the RLS algorithm will be discussed in this section.

In our approach, we use the regularized least square RLS algorithm for two purposes, one is to dynamically assess the domain state of the mobile IoT application for its conformance with user requirements, and the other is to validate the resultant domain state obtained by executing formulated device actions. The SESP specifies the IET for the QoS assessment of the intelligent application with respect to the users' specification, and hence SESP knows the number (n) of the classes IET needed to be classified. We choose to use N binary classifiers to predict datasets relating to each IET class, where N is the number of classes needed to be classified. Our approach only requires O(n) classifiers to determine IET classes [57]. We will use RLS algorithm as our learning algorithm to determine IET classes because the RLS algorithm is simple and requires only relatively small computational power compared to other learning algorithms, such as Random Forest.



Dataset belonging to IET_1
Dataset belonging to IET_2
Dataset belonging to IET_i
DSCs
LV_1
LV_2
LV_i
USC_1
USC_2
USC_n
Categorize DSCs based in their respective LVs and randomize train/test data split
Training DSCs data with respective LVs
Testing DSCs data with respective LVs
LVs of selected DSCs
Selected DCSs
Training Algorithm
TA_i
Provide TAs for testing
TA_1
TA_i
IET
Training of RLS Algorithm
Compare IET with respective LVs of DSCs in test data for accuracy.
Accuracy results
Testing accuracy of TAs

Figure 10: A Technique for Training the RLS Algorithm for Assessing Dynamic IoT Environment

The following steps illustrate the training phase of the assessment algorithm,

Step 5.4.1) Obtain the DSs and their respective LVs generated in Step 4) of our overall approach.

Step 5.4.2) Categorize the DSs using their respective LVs for each IET. Select DSs belonging to one IET class as a positive class and randomly select an equal number of DSs belonging to other IET classes as a negative class. In this case, the trained RLS algorithm will predict the IET of the positive class in the subsequent steps.

Step 5.4.3) Perform a random split of the categorized DSs, and use 80% of split as training data and remaining 20% as the test data.

Step 5.4.4) Perform cross validations on the training data and iterate the RLS algorithm to generate the trained RLS algorithms

Step 5.4.5) Use the generated trained RLS algorithms to assess the 20% test data to predict the IET classes for test DSs, which are used to assess the accuracy of the obtained trained RLS algorithms by comparing with known test LVs.

Step 5.4.6) The accuracy values are reported to the DS categorization module. For trained RLS algorithms having unsatisfactory accuracy values, the DSs will be re-categorized and randomly split to improve the accuracy.

Step 5.4.7) Repeat Steps 5.1) to 5.6) until an accurate trained RLS algorithm is generated for classifying DSs for each of the IET classes.

The trained RLS algorithm will be used for assessing the dynamic IoT environment. The IET classes indicate the conformance of the IoT environment to the users' IoT requirements. In our approach the regularization term that we find more appropriate for the least squared objective function is the l_2 norm. The reason for that being that the l_2 norm forces the coefficients of the least relevant variables in the overall objective function to be very small value, instead of forcing it to zero. In dynamic environments this is very important as some of the least relevant variable may be necessary for adapting the model more effectively in dynamic environments, such as in IoT environments.

5.5 Device State Adaptation and Validation

In this section we will discuss how to formulate the device actions and validate them in a mobile IoT application to achieve user requirements in an efficient manner. The *MDP* planner is located at the *SESP* mobile cloud platform. When the user wants to perform the action formulation and validation, the current *DS* is sent to the mobile cloud for processing. The planner after processing the data sends back the recommended *DS* information after validation along with the action to be executed by the *CM*. We choose the *MDP* planner in our approach because it is probabilistic and reflects the users' behavior more accurately. Since the *CM* is most likely to be installed in a smart phone or other devices with limited computational capacity the planning algorithm has to be executed remotely in *SESP* mobile cloud. This allows us to predict with greater accuracy the target *DS* and actions that reflects user behavior. Light weight non-probabilistic and strictly deterministic planners cannot reflect user behavior and requires continuous re-planning since most of plans may not be liked or followed upon by the user. Factored *MDP* representations can be used to reduce the state space significantly but this also reduces the degree of freedom of value functions and will increase the *SESPs* cost of finding the optimal solution. Hence these kinds of planning techniques are not suitable for our approach.

The representational model for our *MDP* planner has 4-tuples which can be represented as shown in (7),

$$< DS_t,\ ST,\ A_t,\ LV_{DS_t} > \qquad (7)$$

where DS_t is the domain state at time t, ST is the state transition model as shown in Figure 2, A is the action set and LV_{DS_t} is the cost/reward metric respectively. The cost/reward metric is nothing but the LV for each Domain state, which we had already computed.

$$A_t = < NOC, SWTH > \qquad (8)$$

where A_t is the device actions possible from current domain state at t. The IoT adaptation system has two fundamental actions, NOC (no operation) and SWTH (switch). When NOC action is chosen the IoT domain state remains relatively same and when SWTH action is chosen the domain state of the IoT is changed.

The resultant state of the either of the actions are governed by the state transition probabilities which are generated from the user behavior and preferences. The state transition probabilities is initially assigned by the SESP and later updated based on the observed changes in user behavior(s) and preferences.

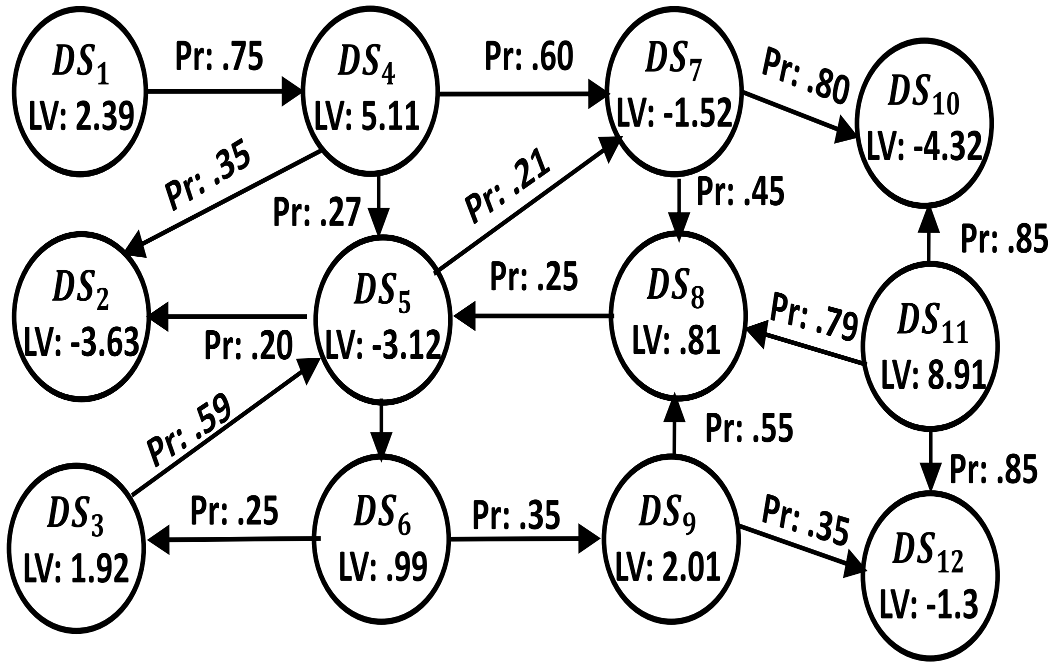


Figure 11: State Transition Model for Action Generation in IoT Application

The state transition diagram, as shown in Figure 11, will be generated by the *SESP*. The edges of the state transition diagram have weights which are probabilistic that represent the likelihood of the user to do a particular action. The initial probabilistic values are assigned using the global user action likelihood metrics, but later they are updated using a gradient descent approach based on *SESP* observation of the users' behavior. In the *SESP* mobile cloud servers a dedicated cache is maintained where a state model with frequently used and most efficient domain states are stored. The edges of the state model are pruned based on their importance using stochastic shielding approximation [58] technique. The threshold value which decides the extent of edge pruning is tuned by the *SESP* based on mobile IoT application. The idea is to reduce the complexity of the planning process by reducing the size of the state model. The state model is dynamic in the sense domain states and edges are brought in and out of the cache copy based on user behavior and probabilistic value ranking.

In our approach we use the mobile IoT application specification provided by the user to formulate the state model. Once state model is generated we initialize the algorithm with random policy (set of actions in a user environment) and then use the backward induction to assess the policy P. The algorithm constantly does a backward induction using a bellman equation [59] and this technique is called Bellman Backup. The policy iteration for an infinite horizon is described using (10), which we use to planning the device actions of a mobile IoT application

$$V_P(DS_t) = LV_{DS_t} + \beta \sum_{DS_{t'}} ST(DS_t, P(DS_t), DS_{t'}) \cdot V_P(DS_{t'}) \quad (9)$$

$$P'(DS_t) = \arg\max_{A_t} \sum_{DS_{t'}} ST(DS_t, A_t, DS_{t'}) \cdot V_P(DS_{t'}) \quad (10)$$

where V_P is the value function of a policy P for a current domain state t, LV is the label values as we had defined before in section 5.3, β is the discount factor which is used to control the learning metric, t' is used to indicate a new domain state neighboring to domain state t.

The brief description on working of the algorithm is as follows,

```
1. Given a policy $P$
2. Loop:
   (a) Evaluate $V_P$ using (10)
   (b) For each $D_i$ in $DS_t$, set improved policy using (10)
   (c) Replace $P$ with $P'$, where $P'$ is the new policy
   Until no improving action possible at any state
```

The result of this algorithm is an optimal policy which specifies what device actions need to be initiated in an IoT application to achieve user requirements. Since we do not have maximization function in (9) we vastly reduce the complexity of the algorithm. Once we get the optimal set of action, the resultant *DS* is sent to the *CM* where our trained *RLS* algorithm will assess the *DS* for its effectiveness. Upon validation by trained *RLS* algorithm the actions are executed.

5.6 Evaluation

In this section, we will present an example to illustrate our approach to intelligently adapt device states for an IoT mobile cloud application. Consider the mobile IoT application in a smart car, where the user wants to achieve maximum security and energy efficiency with comfortable environmental circumstances. The security goals include preventing the user impersonation and intrusion attacks. This is possible because in our approach we create the user's profile, which can be used to detect hackers. Let us assume that the user has the following u-things and non u-things in the mobile IoT application of his smart car: smart air-conditioning system (*AC*), smart infotainment systems (*IT*), smart window (*SW*), smart speed controller (*SC*), smart refrigerators (*RF*), food heaters (non u-thing) (*FH*), vipers (non u-thing), smart phones (*SP*), computers (*CM*) and tablets (*TB*).

Step 1) The user provides the *SESP* the functional and quality requirements of the smart car mobile IoT application, including "Energy Savings" and "security".

Step 2) The SESP identifies the list of u-things, non u-things and the state information for each of the above devices from the device registry which is available from the manufacturers of the devices.

Step 3) The SESP analyzes the attributes of the states of the devices to generate *UV* and *IM*. For example, in the smart air conditioning system, SESP can derive the ranking of importance of the device's states by analyzing the attributes, such as, blower, time and temperature. As discussed in Section 5.3 based on the ranking of device's states, the SESP can generate the *UV*. Table 1 shows the UV and IM values of the u-things and non u-things of the mobile IoT application.

The impact factor of each device in the smart car mobile IoT application is determined by the *SESP* based on the effect each device has on the intelligent application.

Table 2: *UV* and *IM* Values for IoT devices

	State 1	State 2	State 3	State 4	State 5	IM
AC	-3.5	-1.5	-0.2	1.5	4.5	9
IT	-4.5	-3	0.5	4	4.5	4
SW	-4.2	-2	4	0	0	5
SC	-4	4.9	4.9	0	0	8
RF	-3.5	-1.5	3.5	4	4	6
SP	-4	-2.5	1	2	4	2
CM	-2.5	0.1	2	4	0	6
TB	-2.5	-1	3	4	4.9	3
FH	-1					5.5
VP	-2					4

In Table 2, the lightings has higher than normal impact factor even though it has relatively small power consumption because of the fact that it also radiates heat to the environment which has a direct impact on the environment temperature.

Step 4) The SESP calculates the LV using equation (5) for the DSs to be used as training data. Table 2 shows the *LV* for some of the *DSs*,

Table 3: The LV of device state combinations of devices in IoT application

No	DS	LV
1	AC_{S_3}, IT_{S_2}, SW_{S_1}, SC_{S_2}, RF_{S_1}, FH_{DS}, VP_{DS}, SP_{S_2}, CM_{S_1}, TB_{S_3}	-4.11
2	AC_{S_4}, IT_{S_2}, SW_{S_3}, SC_{S_2}, RF_{S_2}, FH_{DS}, VP_{DS}, SP_{S_2}, CM_{S_1}, TB_{S_3}	2.72
3	AC_{S_3}, IT_{S_3}, SW_{S_3}, SC_{S_2}, RF_{S_2}, FH_{DS}, VP, SP_{S_3}, CM_{S_2}, TB_{S_3}	5.39
4	AC_{S_4}, IT_{S_3}, SW_{S_3}, SC_{S_3}, RF_{S_3}, FH_{DS}, VP_{DS}, SP_{S_4}, CM_{S_4}, TB_{S_4}	12.22
5	AC_1, IT_{S_2}, SW_{S_1}, SC_{S_1}, RF_{S_4}, FH_{DS}, VP, SP_{S_1}, CM_{S_3}, TB_{S_1}	-8.95

In this example, based on the energy savings and security of the users' smart car mobile IoT application, the *SESP* identifies the following three *IET* classes,

1. Marginal class, indicate that the devices in this class satisfy certain user specifications, but may need correction to satisfy all the user's application specifications.
2. Satisfactory class, indicate that the devices in this class satisfy all the user's application specifications
3. Unsatisfactory class, indicate that the devices in this class do not satisfy the user's application specifications.

Based on Table 3, the SESP can classify the DSs as follows: #2 belongs to Marginal Class, #3 and #4 belong to Satisfactory Class and #1 and #5 belongs to Unsatisfactory Class according to the values in the *LV* column.

Step 5) The SESP uses the categorized DSs in *Step 4)* as the training data to generate the trained RLS algorithms using our approach discussed in Section 5.

Step 6) The mdp planner on the *SESP* is used for action formulation and trained *RLS* algorithm is used for action validation before actions are sent to mobile IoT application.

The following steps occur in the users' intelligent IoT application:

Step 7) The *SESP* provides/installs a controller module (CM) in the user's smart car mobile IoT application to monitor all u-things states. At each preset interval, the u-things send their state information to the *CM*, which in turn will form the *DSs* and provide them to the trained *RLS* algorithm.

Step 8) The trained *RLS* algorithm uses each of *DS* at every time interval to generate *IET* classes.

Step 9) The *SESP* incorporates adaptive mechanisms where *CM* sends state information to *SESP* through mobile cloud infrastructure for action formulation used for adjusting the necessary u-things actions to achieve the user's expected environment based on the *IET* classes from *Step 7)*. Since the *CM* does not need to control the *FH* and the lights due to the non u-things characteristics specified by the *SESP*, the *CM* only provides the recommendation on the required actions of the non-thing devices to user.

In our approach, the most uncertain and difficult part is the training of the *RLS* algorithm. In this section, we will present the simulation results of our assessment technique using trained *RLS* algorithms because the assessment of the devices in the mobile IoT application is recurring and has a major impact on the effectiveness of our approach.

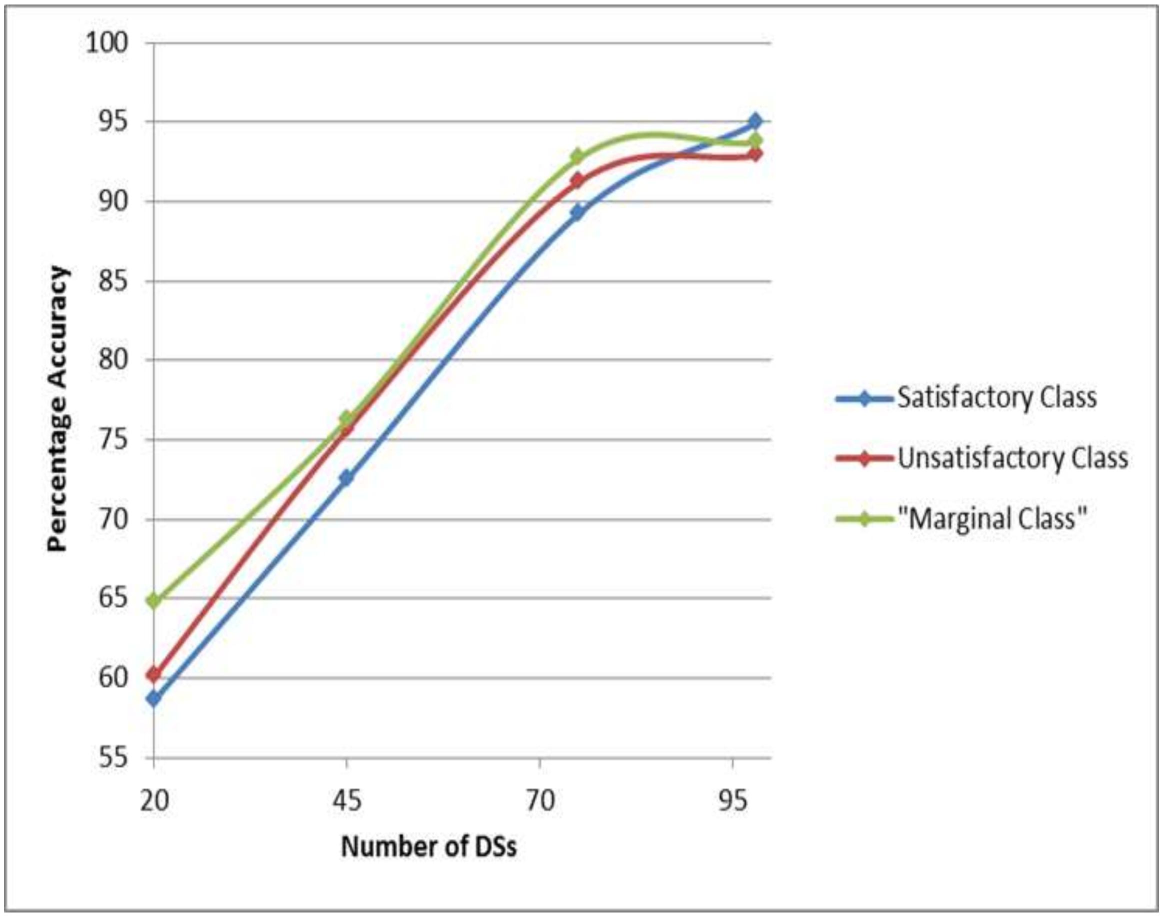


Figure 12: Comparison of accuracies of RLS algorithm for different IET classes

We used the MATLAB to run our simulation of the assessment of the devices in a mobile IoT application using trained *RLS* algorithms. The application consists of 10 devices, with each device having 3 to 5 states. Figure 12 compares the accuracy of the trained *RLS* algorithm in our assessment technique for different *IET* classes. We generated a total of 148 *DSs* and used (5) to calculate their respective *LVs*, some of which are shown in Table 2. We classified the 148 *DSs* into three *IET* tiers based on their *LVs*, and used 80% of the randomly selected data from these *DSs* as the training data and the rest as the test data.

We further partitioned the training data into 10 data sets, and randomly selected a combination for each iteration of *RLS* algorithm run from all possible C_8^{10} combinations, and then run the *RLS* algorithm with initialized parameters. Based on the initial performance of a *RLS* algorithm, the parameters are tuned and the *RLS* algorithm is run with a different combination of datasets. This process continued until such a set of parameters that generated the highly accurate trained *RLS* algorithm was found. The resultant trained *RLS* algorithm was tested with the original test data, which yielded an average trained *RLS* algorithm accuracy rating of 92% with an average precision of the accuracy rating 89%. Figure 3 shows how the accuracy for each IET classification changes with various *DSs*. We observe that for the dataset of the 148 *DSs*, the average accuracy ratings of the trained *RLS* algorithm become stabilized as they approach 75 *DSs*, which indicates that the trained *RLS* algorithm is ready for deployment in the user's mobile IoT application.

5.6 Further Improvements

In this chapter, we have presented an intelligent planning approach to developing cloud-based mobile IoT applications using the *RLS* learning algorithm and *MDP* planner, with only one-to-one interaction between each u-thing and one of the controller modules. Our simulation results showed that the accuracy and precision metrics are relatively high for our assessment technique. We plan to extend our approach to dynamically update the trained *RLS* algorithm for better adapting the users' intelligent IoT applications in real world environments. In addition, we also plan to explore the possibility of using customized factored *MDP* planners to reduce the computational requirements while maintaining high plan optimality rate. Usage of factored *MDP* will also enable us to move the planning to user mobile IoT applications which will reduce transmission overlays, and thus improve our approach

Chapter 6

CONCLUSION AND FUTURE WORK

In this dissertation work we have presented our comprehensive approach to securing ubiquitous smart devices in IoT environment with probabilistic user behavior(s) prediction. My work in this dissertation mainly focused addressing two research problems: (1) how to continuously learn and authenticate legitimate user(s) only using the finger gestures on their touch-based smart device in the background without requiring the legitimate user(s) to provide their finger gesture data intentionally, and (2) how to efficiently configure IoT devices through controller device(s), in conformance with the probabilistic human user behavior(s) and preferences, to effectively adapt IoT devices to the changing environment. Both of our above research works were evaluated with simulated and real-world data, and the results have shown that our approaches were able to effectively address the research problems.

6.1 Summary of contributions

In Chapter 4, we presented an adaptive approach for continuously authenticating legitimate user on a touch screen based smart device (or controller device). Specific contributions of our approach include the following algorithms: 1) an algorithm to estimate the context of smart device usage, 2) an algorithm to estimate degree of importance of each cell/state on the user's smart device touch screen, and 3) a technique that uses Markov Decision Process (MDP) algorithm to continuously authenticate users' with computed user gesture model. The evaluation results for our continuous user authentication approach showed that our approach was able to accurately continuously authenticate legitimate user(s) with acceptable error rates. In Chapter 5, we presented an effective approach for intelligent planning of device action in IoT applications. Our approach is designed to work for both individual user and users as a group. Specific contributions of our approach include the following algorithms: 1) a state analysis technique for the service provider, 2) learning algorithm for dynamic IoT application assessment, and 3) a technique that uses MDP (Markov Decision Process) planning to generate efficient IoT device action plans. Our simulation results showed that the accuracy and precision metrics are relatively high for our dynamic assessment technique.

6.2 Future research directions

Effective automation of decision making in complex systems require significant relaxation of the traditional and unsustainable assumptions such as complete state observability and simplistic contextual inputs. Such relaxations should reflect the real world situations such as partial observability and detailed contextual inputs. Much research has been done in this area of partial observability and detailed contextual inputs. Current techniques that handle such complex decision making systems function under unsustainable assumptions, such as complete state observability and simplistic contextual inputs, which were made to make such systems practical by reducing computation complexity. Furthermore, many existing techniques for partially observable environments such as partially observable markov decision process (POMDPs) have serious limitations.

Such systems use system snapshots to estimate the current system state with simplistic contextual inputs, which limits the effectiveness since system snapshots of partially observable environment do not contain adequate information to accurately estimate current system state. In order to greatly improve the estimation accuracy of the current system states and improve automation of complex decision making systems in partially observable environments, including ones for security systems, I intend to perform further research to modify and apply recurrent neural networks (RNNs) [60, 61] with Hessian-free optimization.

REFERENCES

[1] Gartner smart devices forecast on future growth (2014). http://www.gartner.com/newsroom/id/2839717. Accessed on 15th August 2016.

[2] Survey on the smart device shipment, "http://techcrunch.com/2013/04/04/gartner-2012-2017-devices-forecast/", Accessed on May 15, 2016

[3] Erman Ayday and Sridhar Rajagopal, "Secure, Intuitive and Low-Cost Device Authentication for Smart Grid Networks", (2011) The 8th Annual IEEE Consumer Communications and Networking Conference - Special Session on Smart Grids – Emerging Services and Networks, pp: 1161- 1165.

[4] R. S. Gutzwiller, S. M. Hunt and D. S. Lange, "A task analysis toward characterizing cyber-cognitive situation awareness (CCSA) in cyber defense analysts" 2016 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), San Diego, CA, 2016, pp. 14-20.

[5] New York Times report on recent cyberattacks on big businesses (2015). http://www.nytimes.com/interactive/2015/02/05/technology/recent-cyberattacks.html?_r=0. Accessed on 15th August 2016.

[6] Healthcare's Model Approach to Critical Infrastructure Cybersecurity. (2014). https://hitrustalliance.net. Accessed on 15th August 2016.

[7] Wikipedia source on multi-factor authentication. (2016). "http://en.wikipedia.org/wiki/Multi-factor_authentication", Accessed on May 15, 2016

[8] How Secure are Numeric Passwords. (2015). "http://blogs.cisco.com/security/how-secure-are-numeric-passwords", Accessed on May 15, 2016

[9] Milind Tambe, Manish Jain, James Adam Pita, and Albert Xin Jiang. "Game theory for security: Key algorithmic principles, deployed systems, lessons learned." Proceedings of Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on, pp. 1822-1829. IEEE Press, 2012.

[10] Alecsandru PĂTRAŞCU, and Emil Simion. "Game theory in cybersecurity defence." Proceedings of Electronics, Computers and Artificial Intelligence (ECAI), 2013 International Conference on, pp. 1-6. IEEE Press, 2013.

[11] A. Serwadda, and V. V. Phoha. "When kids' toys breach mobile phone security." Proc. ACM SIGSAC conference on Computer & communications security, (2013), pp. 599-610.

[12] A. Buduru and S. Yau, "An Effective Approach to Continuous User Authentication for Touch Screen Smart Devices", Proceedings of IEEE International Conference on Quality, Reliability and Security (QRS), Vancouver, Canada, Pages: 219-226, August 2015

[13] S. Yau and A. Buduru, “Intelligent Planning for Developing Mobile IoT Applications Using Cloud Systems”, Proceedings of IEEE 3rd International Conference on Mobile Services, Anchorage, Alaska, Pages: 55-62, June 2014

[14] Jianhua Ma, Laurence T. Yang, Bernady O. Apduhan, Runhe Huang, Leonard Barolli, Makoto Takizawa and Timothy K. Shih, “A Walkthrough from Smart Spaces to Smart Hyperspaces towards a Smart World with Ubiquitous Intelligence”, Proceedings of the (2005) 11th International Conference on Parallel and Distributed Systems. Pp: 370 -376, Vol. 1, DOI: 10.1109/ICPADS.2005.60

[15] J. R. Wallrabenstein, "Practical and Secure IoT Device Authentication Using Physical Unclonable Functions," 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, 2016, pp. 99-106.

[16] S. Chakrabarty and D. W. Engels, "A secure IoT architecture for Smart Cities," 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, 2016, pp. 812-813.

[17] J. L. Espinosa-Aranda, N. Vallez, C. Sanchez-Bueno, D. Aguado-Araujo, G. Bueno and O. Deniz, "Pulga, a tiny open-source MQTT broker for flexible and secure IoT deployments," Proceedings of 2015 IEEE Conference on Communications and Network Security (CNS), Florence, 2015, pp. 690-694.

[18] Y. Niu, and C. Hao "Gesture authentication with touch input for mobile devices" Security and Privacy in Mobile Information and Communication Systems, Springer, Berlin Heidelberg, 2012, pp. 13-24.

[19] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication", IEEE Trans. Information Forensics and Security, (2013), Vol: 8(1), pp.136-148.

[20] L. Li, X. Zhao, and G. Xue. “Unobservable reauthentication for smart phones”, Proc. the 20th Network and Distributed System Security Symposium, (2013), Internet Society.

[21] J. Muncaster, and M. Turk. "Continuous multimodal authentication using dynamic Bayesian networks", Proceedings of 2nd Workshop Multimodal User Authentication, (2006).

[22] T. Sim, S. Zhang, R. Janakiraman, and S. Kumar. "Continuous verification using multimodal biometrics" IEEE Trans. 29th Pattern Analysis and Machine Intelligence, Vol: 4, (2007), pp: 687-700.

[23] W. Shi, J. Yang, Y. Jiang, F. Yang, and Y. Xiong. "Senguard: Passive user identification on smartphones using multiple sensors", Proc. IEEE 7th International Conference on In Wireless and Mobile Computing (WiMob), (2011), pp. 141-148.

[24] Dan Xie Roderic A. Grupen Allen Hanson, “Context-Aware Search using Cooperative Agents in a Smart Environment”, Workshop on Applications of Computer Vision (WACV), 2009 , pp: 1 – 6,

[25] Lei Tangab, Xingshe Zhoub, Christian Beckerc, Zhiwen Yub, Gregor Schielec, “Situation-based Design: a Rapid Approach for Pervasive Application Development”, (2012) 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing, pp 128-135,

[26] W. P. Liao, Y. Z. Ou, Chu, E. T. H., C. S. Shih, J. W. S. Liu, “Ubiquitous Smart Devices and Applications for Disaster Preparedness”, (2012) 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing, pp 764- 770

[27] Grzegorz Lehmann, Andreas Rieger, Marco Blumendorf, Sahin Albayrak, “A 3-Layer Architecture for Smart Environment Models”, Pervasive Computing and Communications Workshops (PERCOM Workshops), (2010) 8th IEEE International Conference on , pp: 636 – 641

[28] Yue-Shan Chang, Yung-Shuan Hung, Ching-Lung Chang, Tong-Ying Juang, “Toward a NFC phone-Driven Context Awareness Smart Environment”, Ubiquitous, Autonomic and Trusted Computing, (2009). UIC-ATC '09. Symposia and Workshops on , pp: 298 – 303

[29] Ilche Georgievski, Tuan Anh Nguyen and Marco Aiello, “Combining Activity Recognition and AI Planning for Energy-Saving Offices”, (2013) IEEE 10th International Conference on Ubiquitous Intelligence, pp: 238-245.

[30] Claudio Faris, Luci Pirmez, Falvia C. Delicato, Henrique Soares, Igor L. dos Santos, Luiz F. R. C. Carmo, “A control and decision system for smart buildings”, (2013) IEEE 10th International Conference on Ubiquitous Intelligence, pp: 256-261.

[31] Brian D. Ziebart, Andrew Maas, J.Andrew Bagnell, and Anind K. Dey, “Human Behavior Modeling with Maximum Entropy Inverse Optimal Control”, (2009), Proceedings of AAAI Spring Symposium on Human Behavior Modeling, 92-97.

[32] W.C. Moody, Hongxin Hu, A. Apon, "Defensive maneuver cyber platform modeling with Stochastic Petri Nets,", Proceedings of 10th International Conference on Collaborative Computing (CollaborateCom 2014), October 22-25, 2014.

[33] Alex Pentland, Andrew Liu, “Modeling and Prediction of Human Behavior”, Neural Computation 11, pp: 229–242 (1999)

[34] Green, Leonard, and Joel Myerson. "A discounting framework for choice with delayed and probabilistic rewards." Psychological bulletin 130, no. 5 (2004): 769

[35] Pomerol, Jean-Charles. "Artificial intelligence and human decision making." European Journal of Operational Research 99, no. 1 (1997): 3-25.

[36] Claudine Conrado and Patrick de Oude, "Scenario-based reasoning and probabilistic models for decision support.", Proceedings of 17th IEEE Information Fusion (FUSION), July, 2014, pp. 1-9.

[37] Markovian Property. (2016). https://en.wikipedia.org/wiki/Markov_property. Accessed on 15th August 2016.

[38] Daniel Kahneman, “Thinking, Fast and Slow”, Macmillan Publishing, 2011.

[39] De Martino, Benedetto, Dharshan Kumaran, Ben Seymour, and Raymond J. Dolan. "Frames, biases, and rational decision-making in the human brain." Science 313, no. 5787 (2006): 684-687

[40] Framework for securing BYOD, (2012), “http://www.forescout.com/idc-fs-byod-strategy-white-paper/”, Accessed on May 15, 2016.

[41] L. Li, X. Zhao, and G. Xue. “Unobservable reauthentication for smart phones”, Proceedings of 20th Network and Distributed System Security Symposium, (2013), Internet Society.

[42] Y. Chen, and N. D. Thomas, "Active Learning of Markov Decision Processes for System Verification.", Proceedings of 11th IEEE Conference on Machine Learning and Applications (ICMLA), 2012, vol. 2, pp. 289-294.

[43] K. Hamahata, T. Tadahiro, S. Kazutoshi, N. Ikuko, T. Kazuma, and S. Tetsuo, "Effective integration of imitation learning and reinforcement learning by generating internal reward.", Proceedings of 8th International Conference on In Intelligent Systems Design and Applications, 2008, vol. 3, pp. 121-126.

[44] S. S. Yau, Y. Yin, and H. G. An, “An Adaptive Approach to Optimizing Tradeoff between Service Performance and Security in Service-based Systems”, International Journal of Web Services Research, Vol. 8(2), 2011, pp. 74-91.

[45] S. Russell and P. Norvig, “Artificial Intelligence: A Modern Approach”, 3rd edition, Patience Hall (2009).

[46] R. A. Howard, “Dynamic Programming and Markov Processes”, MIT Press, Cambridge, Massachusetts, 1960.

[47] R. Bellman, “Dynamic Programming”, Princeton University Press, 1957.

[48] M. L. Puterman, “Markov decision processes: Discrete stochastic dynamic programming”, John Wiley & Sons, 1994.

[49] S. I. Ahamed, I. Addo, S. S. Yau, A. Buduru, "A Reference Architecture for Improving Security and Privacy in Internet of Things Applications", Proceedings of 3rd IEEE International Conference on Mobile Services (MS), June 2014, pp. 108-115

[50] Jianhua Ma, “Active Media Technology”, Proceedings of 6th International Conference AMT 2010, volume 6335, pp: 5-5.

[51] S. Yau, Y. Yao, A. Buduru, “An Adaptable Distributed Trust Management Framework for Large-Scale Secure Service-based Systems”, Springer Computing Journal, v(96-10), Pages: 925-949, 2014.

[52] S. Yau, H. An, A. Buduru, “An Approach to Data Confidentiality Protection in Cloud Environments”, International Journal of Web Services Research, v(9-3), Pages: 67- 83, 2013.

[53] Liviu P. Dinu and Radu-Tudor Ionescu, “A Rank-based Approach of Cosine Similarity with Applications in Automatic Classification”, (2012) 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp 260-264.

[54] Anna Huang, “Similarity Measures for Text Document Clustering”, Proceedings of NZCSRSC 2008, April 2008, Christchurch, New Zealand

[55] S. Guha, S. Yau, and A. Buduru, “Attack Detection in Cloud Infrastructures Using Artificial Neural Network with Genetic Feature Selection”, Proceedings of 14th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC), Auckland, New Zealand, Pages: 414 – 419, August 2016.

[56] Stigler, Stephen M. (1981). "Gauss and the Invention of Least Squares". Ann. Stat. 9 (3): 465–474.

[57] Ryan Rifkin and Aldebaro Klautau, “In Defense of One-Vs-All Classification”, (2004) Journal of Machine Learning Research, Vol 5, pp 101-141.

[58] Deena R Schmidt, Peter J Thomas, “Measuring edge importance: a quantitative analysis of the stochastic shielding approximation for random processes on graphs”, (2014), The Journal of Mathematical Neuroscience, Vol. 4:6, pp:1-52.

[59] Daphne Koller and Ronald Parr, “Policy Iteration for Factored MDPs”, (2000) Proceedings of the 16th International Conference on uncertainty in artificial intelligence, pp 326-334.

[60] Vu Pham, Theodore Bluche, Christopher Kermorvant, and Jer´ ome Louradour, “Dropout improves Recurrent Neural Networks for Handwriting Recognition”, (2014), Proceedings of 14th International Conference on Frontiers in Handwriting Recognition, pp: 285-290.

[61] Ilya Sutskever, “Training Deep and Recurrent Networks with Hessian-Free Optimization”, 2012, Ph.D Thesis, Univ Toronto

[62] Yimin Chen, Jingchao Sun, Rui Zhang, and Yanchao Zhang, “Your song your way: Rhythm-based two-factor authentication for multi-touch mobile devices,” IEEE International Conference on Computer Communications (INFOCOM’15), Hong Kong, China, April 2015. pp: 2686-2694

BIOGRAPHICAL SKETCH

Arun Balaji Buduru is a Ph.D. candidate in Computer Science specializing in Information Assurance at Arizona State University, and a member of the ASU Information Assurance Center. He received his B.E. degree in Computer Science in 2011 from Anna University at Chennai, India. His research interests include reinforcement learning, cyber security and stochastic planning. He has worked as a research intern as part of Cisco's IoT Architecture Group. His current research focuses on adaptive user re-authentication on touch-based devices, predicting security breaches in critical cyber infrastructures, and user behavior based adaptive IoT device reconfiguration. His research work has been published in top-tier conferences including CLOUD, QRS, MS. He also served as volunteer and reviewer for multiple conferences and journals including IEEE CLOUD, ACM CCS, Transactions on Services Computing.