ACTIVE AUTHENTICATION USING BEHAVIORAL BIOMETRICS AND MACHINE LEARNING

by

Ala'a Arif El Masri

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_____	Dr. Harry Wechsler, Dissertation Director
_____	Dr. Mihai Boicu, Committee Member
_____	Dr. Charles Snow, Committee Member
_____	Dr. Jim X. Chen, Committee Member
_____	Dr. Stephen Nash, Senior Associate Dean
_____	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering
Date:_____	Spring Semester 2016 George Mason University Fairfax, VA

Active Authentication Using Behavioral Biometrics and Machine Learning

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Ala'a Arif El Masri
Master of Science
University of North Carolina at Charlotte, 2006
Bachelor of Science
Coastal Carolina University, 2002

Director: Harry Wechsler, Professor
Department of Computer Science

Spring Semester 2016
George Mason University
Fairfax, VA

Copyright: 2016, Ala’a Arif El Masri
All Rights Reserved

DEDICATION

For my parents Arif and Eman El Masri who never stopped believing in me. It is because of your unconditional love and unwavering support I continue to thrive; and for that, I am forever grateful.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my Dissertation Director Dr. Harry Wechsler. Your constructive critique, technical guidance and continued advice helped make this possible. Your invaluable supervision was instrumental to my success.

I would also like to express my gratitude to my former PhD advisor, Dr. Brent ByungHoon Kang, who even after leaving GMU continued to encourage me and support my research. I appreciate all that you have done for me to make this possible.

A special thanks is extended to Dr. Peter Likarish whose technical expertise and research contributions helped bring this work to fruition.

Another special thanks goes out to my PhD Committee Dr. Mihai Boicu, Dr. Charles Snow and Dr. Jim X. Chen for dedicating the time to provide valuable feedback and help shape this dissertation.

I would also like to thank my dear brother, Emad El Masri for his unconditional support. You will always be the shoulder I lean on.

A big thank you goes to my best friend Salvatore DeGennaro and my sister-in-law Wafa Al-Arayed for being there for my family and never hesitated to lend a helping hand.

I am also grateful for my three children Ammar, Mariam and Eman. I am so sorry for all the time I have spent working on this and not with you. You are the light of my eyes and the beats of my heart. And like Mom always says, we love you guys more than life itself.

Last but not least, I owe a big token of gratitude to my beloved wife, Dr. Dalal Al-Arayed. Thank you for believing in me when others didn't, encouraging me when I felt down and pushing me to achieve my best. I appreciate all you have done for our family, for picking up my slack when I was falling behind, for being a friend, a wife and a mother, but more than anything else, thank you for your patience.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

AdaBoost..... Adaptive Boosting
Area Under the Curve ..... AUC
Automated Teller Machin ..... ATM
Classification and Regression Trees ..... CART
Command line interface CLI
Euclidean Distance..... ED
False Positive ..... FP
False Negative..... FN
Graphical User Interface ..... GUI
Institution of Electerical and Electronic Engineering ..... IEEE
Korea Advanced Institute of Science and Technology ..... KAIST
Linux/Unix ..... *Nix
Microsoft..... MS
National Institue of Standards and Technology..... NIST
Personal Identification Number ..... PIN
Portable Document Format ..... PDF
Privacy, Security and Trust..... PST
Receiver Operating Characteristic ..... ROC
Support Vector Machine ..... SVM
Synthetic Minority Over-sampling Technique ..... SMOTE
True Positive ..... TP
True Negative..... TN
Visual Basic for Applications ..... VBA

ABSTRACT

ACTIVE AUTHENTICATION USING BEHAVIORAL BIOMETRICS AND MACHINE LEARNING

Ala'a El Masri, Ph.D.

George Mason University, 2016

Dissertation Director: Dr. Harry Wechsler

Active, or continuous, authentication is gradually gaining grounds as the preferred method of personal authentication. This is due to the limited nature of standard authentication methods that are unable to guarantee user identity beyond initial authentication. While research in the area of active authentication has explored and proposed various techniques to overcome this problem, we present two new behavioral-based biometric models for active authentication that expand on current research in terms of performance and scope using adaptive user profiles and their dynamics over time. The novel active authentication models are complementary to each other and include: (1) Application Commands Streams Authentication Model (ACSAM) and (2) Scrolling Behavior Authentication Model (SBAM).

ACSAM is based on the commands a user issues while interacting with a GUI-based application. In this model, supervised learning methods including Random Forests,

AdaBoost, Decision Trees and Naïve Bayes are used to predict whether the authenticated user editing a document is legitimate or not. Random Forests bested all the other learning methods considered in correctly identifying the user with an average of 95.43% accuracy (number of users correctly classified), while achieving an F_1-measure and Area Under the Curve (AUC) of 0.953 and 0.735 respectively.

SBAM is based on a user's document scrolling behavior. In this model, both classification and clustering techniques are used to authenticate the identified user as legitimate or not. For classification, supervised learning methods including Random Forests, AdaBoost and AD Trees (Alternative Decision Trees) are used. While classification using Random Forests with subsampling yielded an average of 98.24% accuracy, it was biased towards the majority class (imposters). This was evident when examining the F_1-measure for the rare class (legitimate users), which at best achieved 0.50 accuracy. Alternatively, unsupervised learning using K-means clustering was shown to narrow down the possibility that a given scrolling behavior belongs to a particular set of users. Towards that end, two approaches were applied to mitigate the unbalanced authentication aspect. The first approach focused on ranking the users with 58% and 80% of the time the actual user ranked in the top 5 and 10 users, respectively. The second approach focused on feasibility sets (or multiple ID sets) with 83% of the time the actual user within the correct set of possible user profiles.

CHAPTER 1: INTRODUCTION

This research addressed the security task of continuously and actively verifying the identity of an authenticated user beyond initial authentication. This is a critical property of the next generation authentication models and a natural progression of traditional authentication techniques. This chapter introduces the related security tasks, associated challenges and motivation, proposed solutions and novel contribution to this research area.

1.1 Challenges and Motivation

Despite the wide consensus on the vulnerability of password-only authentication, 52% of organizations still employ such policies [1]. The problem with password-only authentication is that the system is dependent on a single and, in many cases, weak factor for authentication. If exploited, the system is breached and its resources are vulnerable to unauthorized access. In a report published by Verizon in 2013, security breach incidents were reported by 19 global organizations that shows that 76% of all breaches investigated were due to the exploitation of weak or stolen credentials [2]. Consequently, there has been a push for organizations to strengthen their authentication policies by adopting two-, and sometimes three- factor authentication, in which the user is required to provide more than one method of identification (two- and three- factor authentication are discussed in details later in section 2.2.2 Multi-factor Authentication).

While multifactor and multimodal authentication models provide robust user identification mechanisms, concerns over verifying the user's identity beyond initial authentication remain in demand and have dominated recent discussion. The problem of ensuring that the active user is still the same user who was initially authenticated led researchers to search for additional means of identification that span the duration of the user's active session and provide continuous authentication. This concept, sometimes referred to as active or continuous authentication, requires that a system attempts to verify that the identity of the authenticated user stays the same throughout his/her active session.

To best demonstrate the need of active authentication models, consider the following scenarios:

- **Compromised Credentials:** Assume that Ammar, a legitimate user, logs into his system by providing valid credentials. While he is logged in, Ammar leaves his computer without logging out or locking his screen. An imposter takes over his active session without a way for the system to detect the intrusion. Similarly, stolen credentials can be used by an imposter to log in to the system and gain access to system resources otherwise available only for the legitimate user. Moreover, relying on password-only authentication can impose the risk of passwords being guessed, especially when strong password policies are not in place. This scenario equates to the majority of security breaches in 2012 (76%) according to the 2013 Data Breach Investigations Report by Verizon [2].

- **Insider Cyber Threat:** A definition of insider threat can be borrowed from [3], in which it is defined as a threat that occurs when a trusted entity, someone with the power to violate a set of rules in a given security policy, abuses his/her power. Consider Mariam, a disgruntled employee with valid credentials. Mariam logs into the system, but this time with an ill will towards her employee, decides to corrupt and delete business-critical data. Since Mariam is a legitimate user, a traditional authentication model has no means to detect such adverse actions. According to [2], 14% of data breaches in 2012 were perpetrated by insiders to the organizations.

- **Ubiquitous Computing**: Organization are increasingly adopting mobile technologies that are designed to increase their employees' productivity by keeping them connected to the organizational resources and offering the apps needed to carry out their daily business operations. In fact, a new business model referred to as Bring-Your-Own-Device (BOYD) encourages employees to use their own personal mobile devices for work purposes. However, with the rise of mobile computing, organizations are faced with new security challenges. Due to their size and ubiquitous nature, mobile devices are easy to steal or lose, which increases the risk of sensitive business data falling in the wrong hands. Consider Eman, a legitimate employee that uses her mobile phone to conduct many of her daily business activities (e.g., business email communications). While out for lunch, Eman forgets her phone. A stranger finds it and goes through her email

exposing sensitive client data. According to a recent survey published in 2014 by [4], 68% of reported data breaches since 2010 were due to the loss of theft of employees' mobile device.

- **Malware**: Malware, short for malicious software, is a term used to describe a wide range of software types including computer viruses, worms, Trojan horses, spyware, adware and others that are designed to execute illegitimate and harmful actions against the target system. While how malware is designed to carry out its malicious activities may vary based on its adverse objectives, all exploit a common threat to a system's availability, integrity and/or confidentiality. According to [2], 40% of reported data breaches in 2012 were the result of malware. Both academia and industry continue to explore and propose new methods for detecting such adverse software through the design and implementation of intrusion detection systems. These systems are designed to detect abnormal activities on the host system, which is a core design component of active authentication models today.

In all the above scenarios, if an active authentication model is in place, the system could detect the abnormal behavior, which then could trigger a response such as blocking or deactivating the imposter's access. While the above scenarios by no means represent an exhaustive list of the shortcomings of traditional authentication models, they do showcase the need for active authentication models. However, it is important to emphasize the two main characteristics of an active authentication model: continuous and

non-intrusiveness (these are further defined and discussed in 2.2.3 Active Authentication).

1.2 Thesis Statement

Traditional authentication models fall short of verifying the user's identity beyond the initial authentication. Therefore, active authentication models are needed to continuously and noninvasively ensure that a user is indeed the original authenticated entity throughout her active session.

1.3 Solutions and Methods

Both academia and industry continue to propose, design and implement solutions to the problem of active authentication. The majority of that work relies heavily on biometric technologies and machine learning in its design (both discussed in Chapter 3: Background and Related Work and Chapter 4: Machine Learning). This research explores and proposes two new active authentication models. The first model is based on the sequence of user-issued commands from GUI-based application (see Chapter 6: Application Commands Streams Authentication Model (ACSAM)). The second model is based on user's document scrolling behavior (see Chapter 7: Scrolling Behavior Based authentication Model (SBAM)).

In both models, various machine learning methods have been evaluated and compared including classification and clustering. In classification, predictive modeling is used to identify the class label (*authenticated* vs. *imposter*) of a new observed user/document interaction represented as a feature vector. Decision tress (C4.5) and

Naïve Bayes are evaluated for performance. To improve the model's classification accuracy, ensemble methods are used including Random Forests, AdaBoost and SMOTE (Synthetic Minority Over-sampling Technique). These methods tend to perform better than a single classifier when dealing with small datasets, as such is the case with our datasets. In addition, subsampling techniques are used such as leave-one-out and k-fold cross validation to overcome this same problem.

In clustering, K-means method is used for unsupervised learning with the goal of narrowing down the possibility of reading pattern belonging to a smaller set of predefined user profiles. Two approaches were applied. The first approach focused on ranking the users into the top 5 and 10 possible users for a given working session. Here, a basic simple ranking technique is applied in which a user profile is represented as a set of centroids. Given a new working session, the minimum Euclidean distance to a user's profile is calculated (for details see section 7.3.2.3.1 Approach 1: Top 5/10 Ranking (Singleton Sets)). The second approach focused on feasibility sets (or multiple ID sets). Here, possible users are filtered by their profile standard error. A per-profile distance threshold is calculated based on the average distance and standard deviation between the different reading sessions in each profile and the centroids (for details see section 7.3.2.3.2 Approach 2: Feasibility Sets (Multiple ID Sets)).

Both models can be abstracted and separately applied to a general authentication protocol with four distinct phases: observation, creation/update, comparison and decision phases. This general protocol is designed for user/document interaction; the focus of this

research (we define the nature of this interaction later when discussing the actual models). The following is a description of each phase:

- **Observation phase**: During this phase, the system non-intrusively monitors the user's interaction with a document and collects predefined data points known to provide discriminative properties.
- **Profile creation/update phase**: During this phase, the system references a user profile repository. The goal is to determine if a user profile exists for the current user based on previous interaction with documents. If no profile is found, a new user profile is created and updated with the data being collected. If a user profile is found, it is retrieved and updated with the data being collected.
- **Comparison phase**: During this phase, the system attempts to make a determination whether the user is indeed who he/she claims to be by comparing the newly collected data with the saved profile. Machine learning algorithms can be applied to make this determination.
- **Decision phase**: Finally, the system makes a decision whether to allow the user to continue interacting with the document or deny access. This is a binary classification problem in which the algorithm employed attempts to classify the current interaction as one belonging to the authenticated user or not. In the case where access should be denied, the system can trigger various protective actions based on the policy in place such as locking the system, sending alerts to system administrators and/or prompting the user to re-authenticate.

In cases where the authenticated user is wrongfully denied access (False Positive), the system should be able to adapt by updating the user profile with the interaction data points that led to the misclassification; and hence reducing future false positives. Therefore, the system continues to evolve by learning changes in the user's document interaction behavior. Figure 1 provides an illustration of the proposed general active authentication protocol discussed.

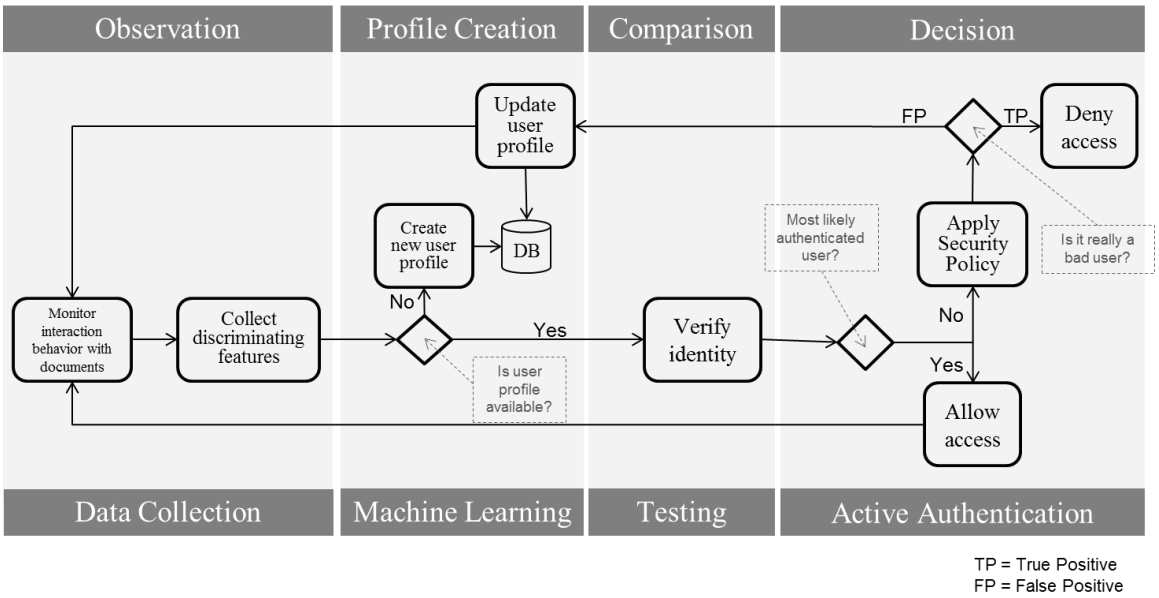

Figure 1: A general active authentication protocol

1.4 Performance

In ACSAM, classification with ensemble learning methods including Random Forest and AdaBoost both perform well. Overall, Random Forests bested other classifiers

in terms of accuracy (average number of users correctly classified), F_1-measure and AUC with 95.43%, 0.943 and 0.735 respectively. This is due to the intrinsic characteristics (e.g. randomness and sampling/bagging) of Random Forests, which cope better with the varying nature of individual user profiles. This level of optimal performance is in line with other research which has found that ensemble learning, either through boosting or bootstraps aggregating (bagging) is flexible enough to represent complex hypothesis functions that are difficult to learn with a single classifier [5]. In addition, bagging (used by Random Forest) can reduce the concern that the learned model may over fit the dataset.

However, to reduce the false negative rate (authenticated users are misclassified as *imposters*), the majority class was subsampled in the training set to 10% of its total size, bringing the number of sessions in the authenticated and other classes much closer to an even split. The results show that subsampling the majority class makes all the classifiers more sensitive to the minority class. Although the average percent correct rate declines by 9.67% for the Random Forest algorithm, the average AUC increases to 0.746. The relative significance of AUC comes from the derivation and interpretation of ROC where decision-making is a function of setting thresholds on similarity distances to trace the ROC curve. Therefore, subsampling has made the classifier far more sensitive to classifying authenticated sessions as *authenticated* (true positives have increased). Subsampling the majority class thus results in a trade-off: a substantially lower false negative comes at the cost of potentially confusing additional *imposter* sessions as *authenticated* (a 23.27% increase in false positives).

In scrolling behavior based active authentication model, classification is neutral at best. Random Forests with subsampling achieves a 98.24% average percent correctly classified and an F_1-measure of 0.50 of the authenticated class suggesting a random guess when it comes to determining the class of an authenticated user. Clustering with K-means offers more promising results with 83.5% success rate in associating a new profile with its original user.

1.5 Research Contributions

The contributions of this research come in the novel methods presented, the tools developed, the adoption of our tools and protocols by other research teams and the published literature. The subsequent sections present a summary of these contributions.

1.5.1 New Active Authentication Models

Two new active authentication models are presented: Application Commands Streams Authentication Model (ACSAM) and Scrolling Behavior Authentication Model (SBAM). The first model, ACSAM, builds user profiles from user-issued commands when interacting with a GUI-based application. We demonstrate that discriminative behavioral and cognitive biometric signatures can augment user profiles. Previous behavioral models derived from user issued commands were limited to analyzing the user's interaction with the *Nix (Linux or Unix) command shell program. Unlike a command line interface, which limit the users' interaction to the set of commands they recognize, applications that are GUI-based allow for richer user interactions and could reveal a cognitive process that may help to infer a person's knowledge, intentions and,

more importantly, identity. This is not to be mistaken for what research in the area of human-computer interaction (HCI) have introduced thus far. HCI research has explored the idea of building user profiles based on their behavioral patterns when interacting with such graphical interfaces mainly by analyzing the user's keystroke and/or mouse dynamics. However, none had explored the idea of creating profiles by capturing users' usage characteristics when interacting with a specific application beyond how a user strikes the keyboard or moves the mouse across the screen. In addition, capturing and analyzing commands issued through the interaction with GUI-based applications provides inference on particular knowledge, context and intent (task) from such actions. We provide preliminary evidence that such actions are observable and can reveal the intention behind such interaction.

In the second model, SBAM, we describe and present a novel behavioral biometric based on users' document scrolling traits, and then introduce a new method that leverages this unique trait for re-authenticating users. It is important to note that scrolling occurs as the result of using a mouse and/or the keyboard keys. In particular, the model focuses on identifying anomalous scrolling behavior when users interact with protected or read-only electronic documents. This poses a unique challenge due to the minimal user input that can be observed and analyzed for authenticity. Protected documents such as protected Microsoft Word files and Portable Document Format (PDF) files prohibit user input thereby facilitating the focus on observing activities beyond the traditional modification of, addition to and/or deletion of documents' contents. Hence, this method relies only on how those documents are viewed.

This research provides preliminary evidence to the existence of unique patterns related to how users scroll electronic documents. Such patterns are influenced by many factors including the users' purpose or intent; the documents' contents, length, layout, and type; the environment in which such interaction takes place; users' knowledge of the subject; and users' physical condition. While some of these factors can be controlled such as environment and document layout, others are difficult or impossible to control such as users' physical condition. Therefore, the challenge is to find out if a unique reading pattern exists regardless of external factors.

1.5.2 Developed Tools

In addition, we repurpose and improve a Microsoft Word logger program that was developed in previous work by researcher at MITRE Corporation to capture desired features from the user interaction with Microsoft Word document. The logger program is developed using Visual Basic for Application (VBA) and runs in the background without interrupting the user's session when working on a document.

1.5.3 Research Impact

Furthermore, the MS logger program and the experiment protocol were leveraged by a research group from the Korea Advanced Institute of Science and Technology (KAIST). They conducted similar experiments to those we designed for the ACSAM model (see section 6.3 Experimental Design). The methods and results of KAIST research are discussed in section 8.5 KAIST Research.

1.5.4 Publications

Conference papers on both models were presented at the 2014 Twelfth Annual Conference on Privacy, Security and Trust (PST) [6]; and at the 2015 6th International Conference on Information and Communication Systems (ICICS) [7]. Both papers are also published in the Institute of Electrical and Electronics Engineering (IEEE) scholarly research database, Xplore digital library. Although recently published in April 2015, our research paper on scrolling behavior based active authentication has been already cited by other researcher in the academic community [8], [9].

CHAPTER 2: INFORMATION SECURITY AND AUTHENTICATION

This chapter discusses the topic of information security in relation to our research topic of active authentication. It introduces the objectives of information security and how active authentication is applied to uphold and enhance its objectives. It also provides a brief progression of authentication models towards active authentication.

2.1 Information Security

Information security entails the protection of digital information from being misused by ensuring three main principles: availability, integrity and confidentiality. According to the National Institute of Technology and Standards (NIST), these three principles form the basic objective of information security [10] and can be defined as follows:

- Availability: The protection against the unauthorized withholding of information
- Integrity: The protection against the unauthorized manipulation of information
- Confidentiality: The protection against the unauthorized disclosure of information

These principles have become an industry standard for designing and evaluating a security model. To achieve this objective, a security process needs to be in place. Such process usually consists of three main phases: a prevention phase, a detection phase and a response phase [11]. In the prevention phase, security controls are applied to prevent the

unauthorized access to, withholding of, manipulation of and disclosure of information. If an adversary succeeds in exploiting the prevention controls, the detection phase should be able to identify the abnormal activities and hence triggering the response phase (e.g., applying the appropriate security policy). While this is a simplistic overview of a security process, in practice such a process must follow a layered approach to security in which security controls [12] are applied at different network, application and data layers. Such a layered approach can be thought of as an information security framework for preventing, detecting and responding to security threats. One such control is the identification and authentication control to which this research is most applicable.

In the subsequent sections, we will introduce and describe authentication and its various types including single-factor, multi-factor and active authentication. Figure 2 provides a conceptual view of how a security process works on upholding the objective of an information security.

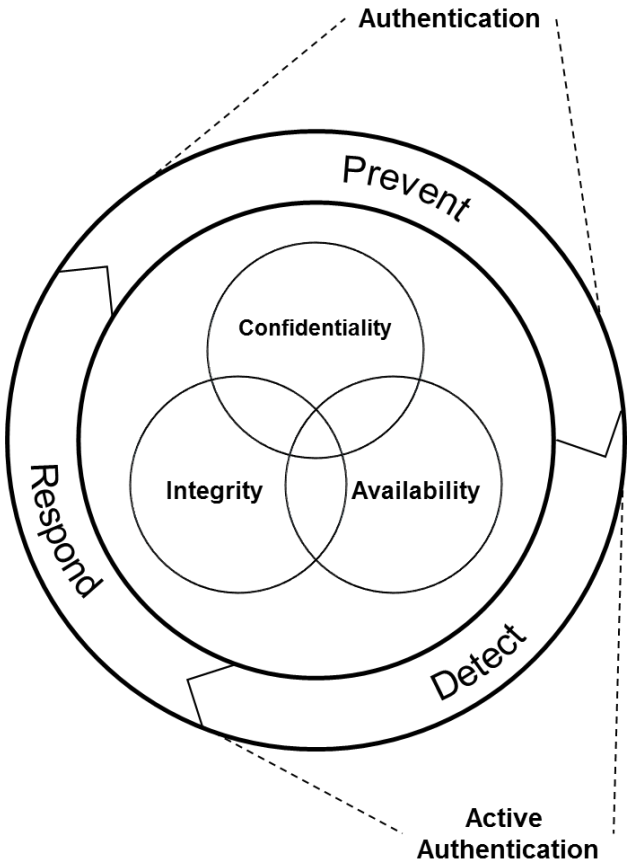

Figure 2: A conceptual view of a security process and its objective

2.2 Authentication

Authentication is the task of verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system [12]. There are numerous methods for implementing an authentication process. These methods can be classified into three main authentication types: knowledge-based, token-based and biometric-based authentication.

In knowledge-based authentication, a user is presumed to know something only he/she knows [13]. Here, the authenticator (e.g., authentication system) and the to-be authenticated entity (e.g., a user) share something that is known only to both of them. For example, both entities know the answer to a previously defined question such as a password or a pin number. This type of authentication is the most common and is sometimes referred to as traditional authentication.

In token-based authentication, a user is presumed to have something only he/she is in possession of. In this scenario, the authenticating entity trusts that only the legitimate user has access to the secret token. For example, the user is in possession of an RSA[1] token or an Automated Teller Machine (ATM) card. Traditionally, this type of authentication is supplemented with a knowledge-based authentication such as a secret password or pin forming what is known as multi-factor authentication.

In biometrics-based authentication, a user is presumed to have a unique physiological or behavioral trait. Here, the authenticating entity attempts to match a previously stored biometric to a new one provided by the to-be authenticated user. If a match is found, the user is presumed to be the legitimate user, otherwise access is denied. This type of authentication is further described in section 3.1 and forms the basis of our authentication models.

These authentication types can be used individually (single-factor authentication) or in combination (multi-factor authentication). The subsequent sections describe this

[1] RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described the algorithm in 1977.

concept further. Figure 3 provides a conceptual view of the multi-factor authentication concept.

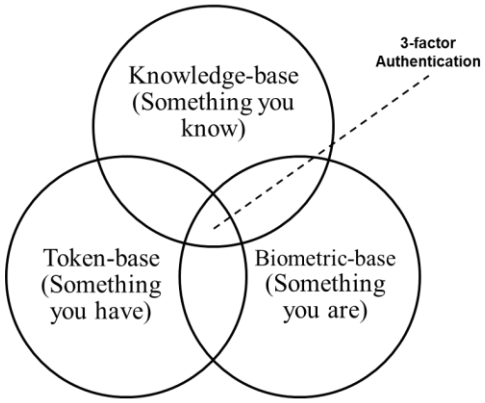

Figure 3: Conceptual view of multi-factor authentication

2.2.1 Single-factor Authentication

In single-factor authentication, only one factor is used to achieve authentication. This factor could apply any of the previously described authentication types including knowledge-based, token-based or biometrics-based authentication. This is the most common type of authentication applied by most websites to authenticate users before accessing their online accounts.

2.2.2 Multi-factor Authentication

In multi-factor authentication, different factors are used to achieve authentication including knowledge-based, token-based or biometrics-based authentication. A classic

example of multi-factor authentication is ATM card. A user needs to provide both the actual ATM card and a personal identification number (PIN) to access his/her fund at a bank automated teller machine.

2.2.3 Active Authentication

The need for verifying the user's identity beyond the initial authentication led to the research of a new authentication method that is capable of continuously validating the authenticity of the originally authenticated entity. This type of authentication is commonly known as active authentication in which the system at various points in time attempts to non-intrusively verify the active user's identity. Consequently, an active authentication system must meet two main properties:

- **Continuous**: The system must continuously verify the identity of its subjects. The verification can be trigger based on time or events. In a time-based model, the system will attempt to verify the identity of the active user based on a predefine time intervals. In an event-based model, the system will attempt to verify the identity of an active user based on the occurrence of an event or a change in state.
- **Nonintrusive**: The system needs to function without interrupting the normal flow of operations by noninvasively monitoring and verifying the user's identity.

To satisfy the nonintrusive property, active authentication models almost always employ biometrics in their implementations. Biometrics allows the authentication model to silently (without intrusion) observe the traits of authenticated users. Section 3.1

Biometric-based Authentication Models introduces the various types biometrics used with active authentication models and the related work.

Active authentication aims to overcome the limitations of traditional one-time authentication methods. These methods are designed to verify a user's identity only at the initial access point. Once authenticated, a user is presumed to be the legitimate user throughout the active session. The system is not designed to re-verify the identity of a user until the current active session has expired and the user attempts to access the system the next time. Consequently, an imposter successfully taking over the legitimate user's session is left undetected. For example, consider a situation where a user logs into a system and then walks away from his/her computer without logging out or locking his screen. An imposter can take over without being detected by the system.

With active authentication in place, the system should be able to detect deviations of expected normal behavior. This normal behavior is usually previously observed and stored in the form of a user profile. As a user interacts with the system his/her previously profiled traits are compared to the occurring ones. Depending on a predefined threshold of confidence, the system makes a decision to whether such new behavior is considered normal or not based on this defined threshold.

Figure 4 illustrates how active authentication models can detect illegal access beyond initial authentication.

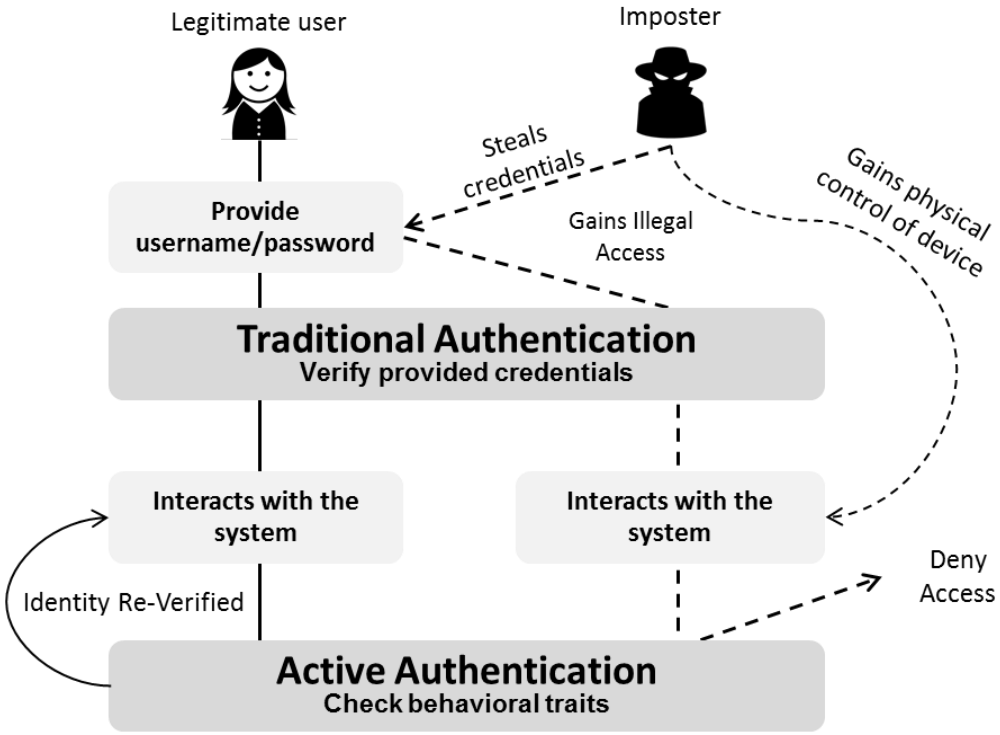

Figure 4: Conceptual view of traditional vs. active authentication rolls in detecting an imposter

CHAPTER 3: BACKGROUND AND RELATED WORK

The concept and the use of continuous or active authentication has gained momentum in recent years for the purpose of verifying the identity of a user beyond initial authentication including interest from the defense sector [14]. Traditional authentication models were designed to authenticate users at the initial stage and, consequently, grant or deny access. However, once a user is granted access there is no way to further attest the authenticity of the user and discover unauthorized use, if any. A continuous authentication system attempts to address this weakness by re-authenticating the user throughout his/her active session. It is important to point out that much of the work in the field of intrusion detection has also utilized behavioral biometrics, and thus lends itself to active authentication. Both active authentication and intrusion detection are about finding imposters engaged in illicit use of resources. The following review of the related work covers both concepts.

3.1 Biometric-based Authentication Models

Various implementations of continuous authentication systems have been proposed and all have naturally employed biometrics as the authentication type. Most of biometric authentication models generally undergo four major stages [13]:

1. **Enrollment stage**: This is the entry point to any biometric system in which the user's desired characteristics (physiological or behavioral) are captured as a set of features that could be used later to uniquely identify the user. This constitutes the registration process and the collected sample is used to create the user biometric profile. The tool used for collecting these samples varies depending on the biometrics being collected (e.g. fingerprint readers, iris scanners, camera, etc.). The number of samples needed for creating the first profile also varies depending on the biometric being used.
2. **Monitoring and Collection stage**: At this stage the, the system collects new samples either actively or at predefined intervals. Samples are collected and the same features are used to create the user profile during the enrollment process are extracted here to be used in the comparison stage.
3. **Comparison stage**: Newly collected samples are compared to the stored user profile or multiple user profiles if the system is to decide who the user is from the pool of preregistered users. This stage can be triggered by an event (e.g. attempting to access a specific resource) or time (e.g. verifying the user's identity at specific time intervals).
4. **Decision stage**: Based on the comparison result a decision is made to whether the user is indeed the legitimate user or an imposter.

Figure 5 illustrates the four general stages of a biometric authentication process.

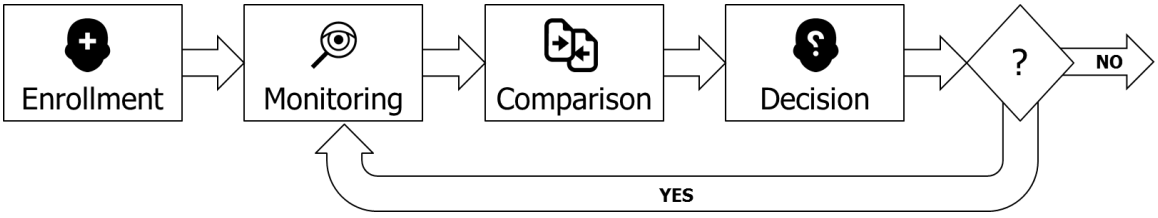

Figure 5: General biometric authentication process

While the related work discussed here is not an exhaustive review of the evolution of biometrics, it does highlight, at various points in time, the research and technologies applied in this field. The subsequent sections provide a review of related work on the use of biometrics for active authentication purposes.

3.1.1 **Physiological Biometrics**

Physiological biometrics rely on an individual's unique physical characteristics such as fingerprints [15], palm prints [16], [17], iris [18], and face [19], [20], [21], among others, to authenticate users. For the most part, authentication systems that employ physiological biometrics draw on external sensors that capture users' physical characteristics. For example, Klosterman and Ganger [22] used a camera mounted on top of a computer monitor to capture the user's face images to determine identity.

It is important to note that the various biometrics techniques are not always designed and implemented independently as recently; researchers [23]–[26] have explored fusing biometric identifiers to create a multimodal biometric system and

improve the identification process. Azzini et al. [26] proposed and tested a multimodal authentication system that utilizes two physiological traits, a fingerprint and face biometrics. The user is initially authenticated by providing his/her username and password. A corresponding template is then retrieved for the face recognition matching phase, which continuously checks the identity of the authenticated user. Once the face biometrics module become unsure of the user's identity, the user is prompted to provide his/her fingerprint for re-authentication. Yap et al. [27], on the other hand, designed a system that utilizes a camera and a biometric mouse that continuously collects both the face and the fingerprint features and fuses them into a composite score that is checked against a threshold to determine authenticity.

3.1.2 **Behavioral Biometrics**

While the above examples rely on external sensors for collecting the biometric data, others have resorted to utilize more traditional input tools such as the keyboard and mouse. In such cases, researches rely on the behavioral characteristics of users. Behavioral biometrics are based on a person's behavior such as how an individual strike a keyboard [28], walks [29], or execute computer commands [30]. For example, many have investigated users' keyboard typing characteristics (i.e., keyboard dynamics) as possible means of continuous authentication. A survey of such methods is provided by Shanmugapriya et al. [31], which discusses the various efforts in this area and compares the different keystroke metrics used. Others [32], [33], [34],[27]–[29] have explored the idea of mouse dynamics as a possible user biometrics. A survey by Revett et al. [35]

provides a detailed review of work in this area and proposes a new one. Ahmed and Traore [36] have combined both keystroke and mouse dynamics for intrusion detection purposes. Others have fused such techniques to create multimodal biometric systems. For example, Grag et al. [37] architect a framework for active authentication that depends on keyboard activity, mouse movement coordinates, mouse clicks, system background processes and user run commands. Unlike our work on application specific command streams, their techniques depend on features collected from the users' interactions with the GUI-based operating system in general and not application-specific features, which is more granular in scope and can be geared towards critical applications that deal with classified material. Such an active authentication technique allows the monitoring of a richer set of behavioral features that reveals a cognitive process that may help to infer a person's knowledge, intentions and, more importantly, identity.

Another popular means of continuous authentication is the analysis of command line lexicons, which is closely related to our ACSAM model for application specific command streams approach to active authentication. For example, Lane and Brodley [38] used the Linux command line prompt to capture a sequence of fixed-length commands and create users' profiles. Detection of normal or abnormal behavior is then calculated by measuring the similarity of new sequence to existing ones.

Marin et al. [39] have also used command lines to detect intruders. They classify legitimate users into categories based on the percentage of commands they issue in a given time period. A total of 5,000 commands per user were collected from a Linux shell for 50 different users. Users' profiles were created from groups of 1,000 commands and,

therefore, each user was represented five times. Their classification methodology employed expert rule to reduce dimensionality, K-means for clustering, a generic algorithm for further dimensionality reduction, and Learning Vector Quantization (LVQ) for refining clusters. Their work has resulted in a classification rate close to 80% (correctly identified users) and misclassification rate less than 20% (incorrectly identified users).

Yeung and Ding [40] have used user profiles created from Unix shell commands and system calls to apply dynamic and static behavioral modeling approaches. The difference between the two approaches is that dynamic models explicitly model temporal variations. The dynamic model is based on hidden Markov model (HMM), while the static model is based on event occurrence frequency. They applied novelty detection instead of classification due to the lack of data representing abnormal behavior for training. Their experiments showed that the static modeling approach yielded better result than the dynamic modeling with the shell commands, achieving at its best a true detection rate (TDR) of 87% and a false detection rate (FDR) of 20%. The static model performed better with shell commands over system calls due to the temporal dependencies between system calls, which the static model was unsuitable to handle.

Schonlau et al. [41] also used Linux shell commands and applied six different classification methods for detecting masqueraders – people who impersonate the authenticated user. Over several months, 15,000 commands per user from 70 users were obtained. Fifty users were selected to be intrusion targets and the remaining 20 user were designated as the masqueraders. Data from the masqueraders was interspersed into the

target data. The first 5,000 commands are used to train the user profiles. For each block of 100 commands, a score is computed and checked against a threshold. Out of the six methods applied the Bayes 1-step Markov performed the best in detecting masqueraders but failed in achieving the desired goal of 1% false alarm rate (FAR). On the other hand, the Uniqueness method (based on command frequency) did well in coming close to the desired FAR but failed to detect masqueraders. All the other methods performed somewhere in between.

Maxion and Townsend [42] have extended the work by Schonlau et al. by revising the experimental method and applying Naïve Bayes, which resulted in a 56% improvement and 1.3% FAR. More recently the same data set used by Schonlau et al. has been used by Traore et al. [30], which used sequential sampling techniques and naïve Bayes classification scheme and yielded 4.28% false acceptance rate (FAR) and 12% false rejection rate (FRR).

Implementations of command line biometrics have, thus far, been limited to the CLI. In this research, we explain how such concept can be extended to other domains and in particular to GUI-based applications that provides the user with more than just command-like interactions. However, in our experiment we only consider command streams issued by users when interacting with MS Word as means of identification. We test our hypothesis with a data set obtained by MITRE Corporation for their earlier work on a recommender system. A description of this data is provided in section 6.1 Dataset.

The most relevant prior work [43] and closest in spirit to our SBAM model for scrolling behavior approach to active authentication introduces a new active

authentication model based on screen fingerprints, in which discriminative visual features are first extracted from screen recordings of the user's interaction with the computer. Those interactions are then classified using Support Vector Machine (SVM) and Adaptive Boosting (AdaBoost), into four possible interaction types: scrolling, typing, mouse-based dragging and window resizing. Each of these interaction types is then tested using soft-margin SVM and k-nearest neighbors to see how well they can identify a user. The features used are limited to using optical flow indicators, the task is identity verification for the whole session rather than active authentication, and the number of users the method is tested on is limited to five users only. Those limitations are removed by our novel full-fledged active authentication method described herein.

CHAPTER 4: MACHINE LEARNING

Machine learning is a scientific discipline that focuses on building prediction models capable of learning from past observations. This is also known as generalization, which is the model's ability to accurately recognize, or predict, new samples based on what it learned from previous observations. Various machine learning methods have been developed to achieve this desired generalization goal including classification (for predicting categorical or nominal values), regression (for predicting continuous or numerical values) and clustering (for predicting group membership). They mostly fall into two broad categories: supervised and unsupervised learning. This chapter describes the differences between supervised and unsupervised learning and the various methods used in this research.

4.1 Methods

4.1.1 Supervised Learning

In supervised learning, the model is trained on a set of samples. Each sample includes input data and its corresponding expected outputs. The model then seeks to deduce an inference function capable of predicting the unknown output of new samples. The general approach for building supervised learning models starts with an induction phase, also known as a training phase, in which the model is trained on a set of samples whose class label is known. This is followed by the deduction phase, also known as a

testing phase, in which the model attempts to predict the class label for new previously unknown samples. Figure 6 illustrates the general approach for building supervised learners.

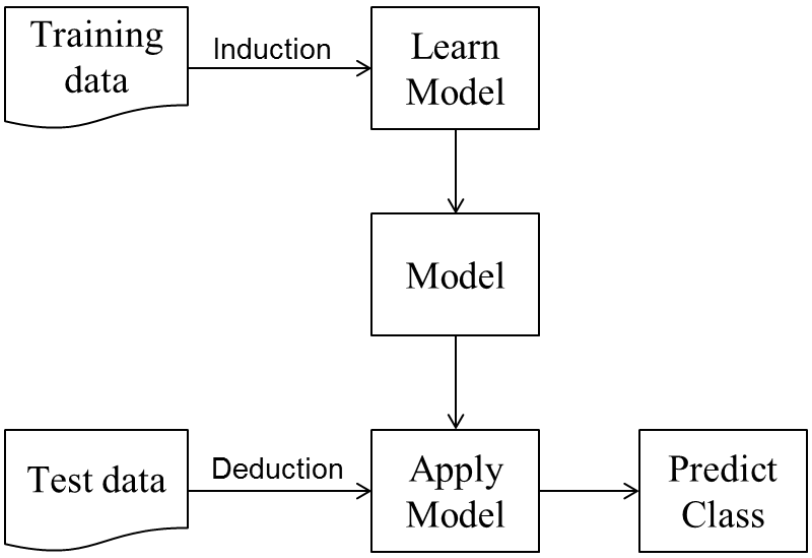

Figure 6: Conceptual View of a General Learning Model

This section discusses the various supervised learning methods used in our research including Decision Tress, Naïve Bayes, Random Forests, AdaBoost and SMOTE.

4.1.1.1 Classification

Classification is the task of assigning an object to one of several predefined classes. As defined by [44, p. 146], classification is the task of learning a target function *f* (also known as a classification model) that maps each attribute set *x* to one of the predefined class labels *y*. Classification models can be used for descriptive or predictive analyses of

data. In descriptive analysis, the task is to understand the relationships between samples in describing a class, while predictive analysis seeks to predict a class label given unknown sample. In this research, classification is focused on predictive modeling in which the task is to identify the class label (authenticated vs. imposter) of a new observed user/document interaction represented as a feature vector.

4.1.1.1.1 Decision Trees (C4.5)

Decision trees are simple yet widely used techniques for predictive modeling. They are constructed of nodes and edges. The tree has three types of nodes: a root node that has no incoming edges and zero or more outgoing edges; an internal node that has exactly one incoming edge and two or more outgoing edges; and a leaf node that has exactly one incoming edge and zero outgoing edges. Root and internal nodes contains the attribute test condition from which a decision to split the tree is based on. The test attribute conditions represent differences in the data characteristics (e.g. average vertical scroll > 0.65) that help the model in separating the samples. Leaf nodes at the end of the tree are assigned a class label (e.g. authenticated vs. imposter). Classification using decision trees can simply be described by the following protocol:

For each new record

1) Start at the root node
2) Apply attribute test condition
3) Based on the outcome of the test in step 2, follow appropriate branch
    3.1) If internal node, then apply new attribute test condition and repeat step 3
    3.2) If leaf node, then stop and assign class label

Constructing an optimal tree is computationally expensive or infeasible due to its exponential nature (numerous trees that can be constructed from a given set of attributes). Therefore, many algorithms have been developed that aim to induce feasible suboptimal yet reasonably accurate trees. Such algorithms relay on a greedy strategy for selecting an attribute that would optimally split the data. A common such algorithm is Hunt's algorithm, which form the basis for many decision trees induction algorithms including C4.5 [44, p. 152]. Hunt's Algorithm can be summarized by the following:

Let D_t denote the set of training records associated with node t

Let y denote the set of possible class labels $\{y_1, y_2, \ldots, y_c\}$

1) If $\forall\, D_t : \in\ y_t$ (all records in D_t belong to the same class label y_t), then t is a leaf node labeled as y_t
2) If $\neg(\forall\, D_t : \in\ y_t)$ (not all records in D_t belong to the same class label y_t), then
    2.1) Select an attribute test condition to split the D_t into smaller subsets
    2.2) $\forall$ test outcome, create a child node
3) Distribute records $\in D_t$ to the children nodes
4) $\forall$ child node repeat starting at step 1

4.1.1.1.2 Naïve Bayes

Naïve Bayes is one implementation of the Bayesian classification methods. These methods, which are based on the Bayesian theorem, seek to model the probabilistic relationships between the attribute set and the class variable. Such models are needed in cases where the relationship between the attribute set and the class variable is non-deterministic (Although the attribute in a given test record might be identical to some training records, the class label cannot be determined with certainty).

For example, consider the training data constructed from our GUI-based Application Command Streams experiment (further discussed in 0). The task is to predict whether a profile belongs to the authenticated user or an imposter based on the count of each command executed from a GUI-based application. Training records are formed using the following attribute set: {*user id, count of command*$_1$*, count of command*$_{2,\ldots,}$ *count of command*$_n$}. A test record may differ from the expected command count learned from the training records and gets assigned the class variable "imposter". However, the test record may truly belong to the authenticated user and the variance of the command count could be related to other factors such as the authenticated user being interrupted while editing the document and hence affecting the sequence in which he/she executed these commands.

Therefore, a classification model such as Naïve Bayes helps in situations where the class variable *Y* has a non-deterministic relationship with the attribute set *X*. In such cases, the model will treat *X* and *Y* as random variables and seek to capture their relationship using the conditional probability *P(Y|X)*, which is also known as the posterior probability for *Y* [44, p. 229]. Similarly, the Naïve Bayes classifier assumes that the attribute set *X* is conditionally independent given the class label *Y*. However, instead of estimating the class-conditional probability for every combination of X, Naïve Bayes will only estimate the conditional probability of each X_i, given *Y*. This approach is suitable in situations where the training set is small since the model can still obtain a good estimate of the probability. The Naïve Bayes classifier calculates the conditional probability using the following formula:

$$P(Y|X) = \frac{P(Y)\prod_{i=1}^{d} P(X_i|Y)}{P(X)} \quad (0.1)$$

The model then uses this formula to estimate the conditional probabilities during the training phase. Subsequently, a test record *X* can be classified by finding the class *Y* that maximizes the conditional probability (since *X* is fixed for every *Y*, it is sufficient to select *Y* that maximizes the numerator only).

4.1.1.2 Ensemble Methods

Ensemble methods are techniques used for improving the model's classification accuracy by aggregating the predictions of multiple classifiers. This is achieved by constructing a model that combines serval base classifiers induced from the training data. The model preforms classification by taking a majority vote on the predictions made by each base classifier [44, p. 276]. This can be further illustrated in Figure 6.

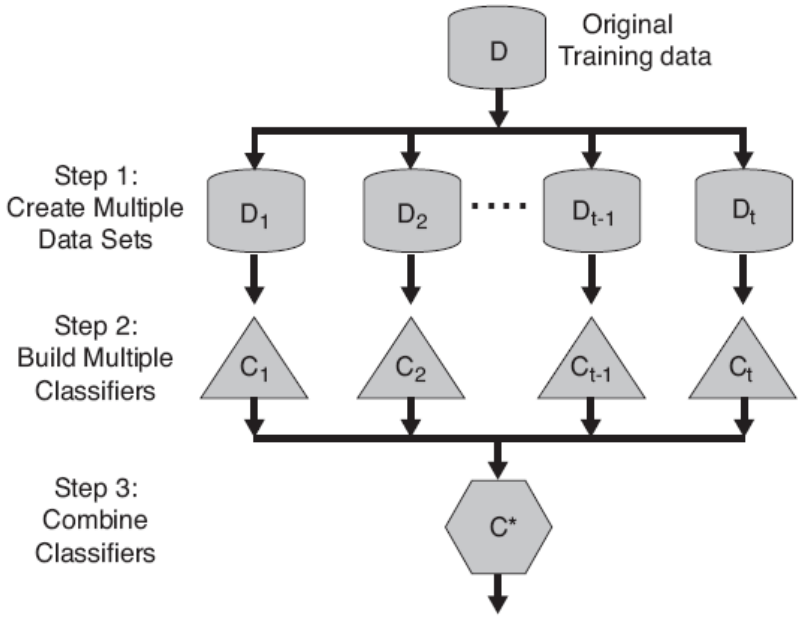

Figure 7: A Logical View of Ensemble Learning Method[1]

Since ensemble methods aim to increase the classification accuracy of a model, it is important to evaluate the reduction of error rate, if any, when compared to single classification methods. However, for ensemble classifier to perform better than a single classifier two conditions must be true: (1) the base classifiers should be independent of each other (their errors are uncorrelated); and (2) the base classifiers should do better than a classifier that performs random guessing.

[1] Figure borrowed from [44, p. 278]

4.1.1.2.1 Random Forests

Random Forests is an ensemble learning method used in both classification and regression tasks. It is constructed from multiple decision trees each generated from an independent random training set. The predictions made by each decision tree are then combined and a majority vote is applied to determine the class label [1, p. 290]. Randomization is an important aspect in Random Forests as it helps in reducing the correlation among the individual decision trees, and hence improves the generalization error (the algorithm's ability to generalize). Bootstrap aggregation, also known as bagging, is used in Random Forest to introduce randomness in selecting the data records.

Performance of Random Forests can be computed from the average performance of the set of classifiers, and is measured probabilistically as the classifier's margin as shown in equation (4.2):

$$margin, M(X,Y) = P\left(\hat{Y}_\theta = Y\right) - \max_{Z \neq Y} P\left(\hat{Y}_\theta = Z\right) \qquad (4.2)$$

Where $\hat{Y}_\theta$is the predicted class X according to a classifier built from random vector θ. The classifier's accuracy increases as the margin increases. Figure 8 provides a logical view of how Random Forests are constructed.

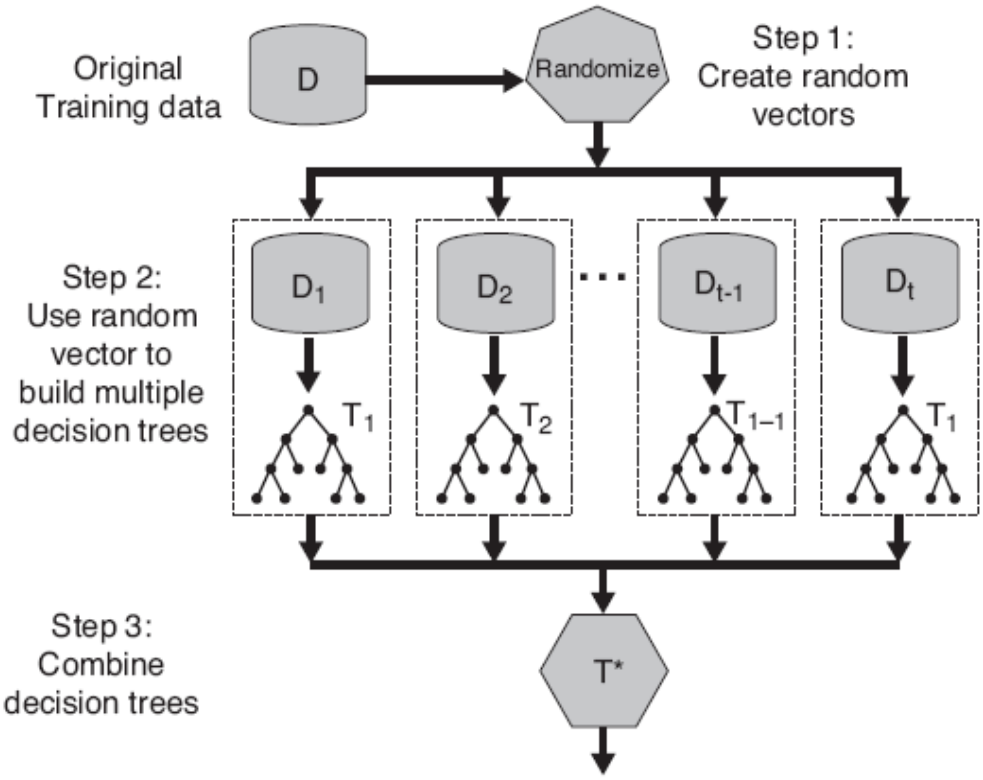

Figure 8: Logical View of Random Forests[1]

4.1.1.2.2 AdaBoost

AdaBoost, short for Adaptive Boosting, is an ensemble learning algorithm that iteratively trains a "weak classifier" on different probability distribution of the training set to boost the performance of a classifier. Unlike Random Forests where the training sets are randomly generated from a fixed probability distribution, AdaBoost adjusts the training set by assigning higher weights to those examples that were hard to or misclassified in previous round. The weak learner, also referred to as the base classifier,

[1] Figure borrowed from [44, p. 292]

in subsequent rounds is forced to focus on the misclassified examples. The final boosted classifier is represented as the weighted sum of the base classifiers obtained from each round [44, p. 288]. Figure 9 illustrates the iterative process of the AdaBoost algorithm.

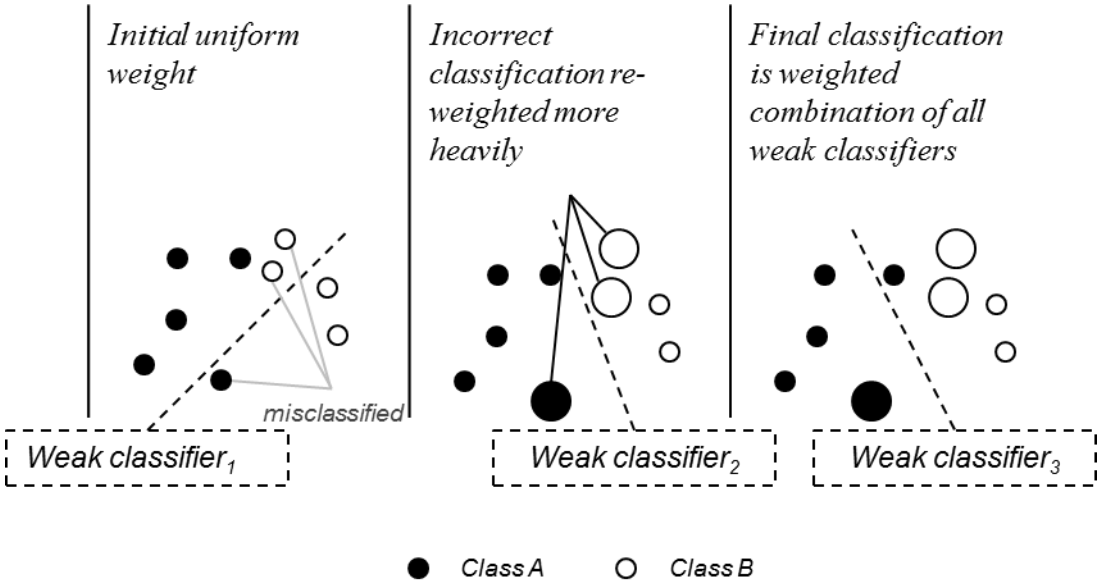

Figure 9: AdaBoost Conceptual View

4.1.1.2.3 Synthetic Minority Over-sampling Technique (SMOTE)

Due to the imbalanced nature of our dataset (the two *Authenticated* and *Imposter* classes are not approximately equally represented), SMOTE technique was applied to decrease the classifier's bias towards the majority class (*Imposter*) and improve the classification performance on the minority class (*Authenticated*). SMOTE is a widely used technique for oversampling imbalanced data where each minority class sample is over-sampled by introducing synthetic examples along the line segments connecting the *k* minority class nearest neighbors. The actual neighbors from the *K* nearest neighbors are

randomly selected depending on the amount of over-sampling required [45]. Figure 10 illustrates how SMOTE selects the synthetic sample from K nearest neighbors of a given sample.

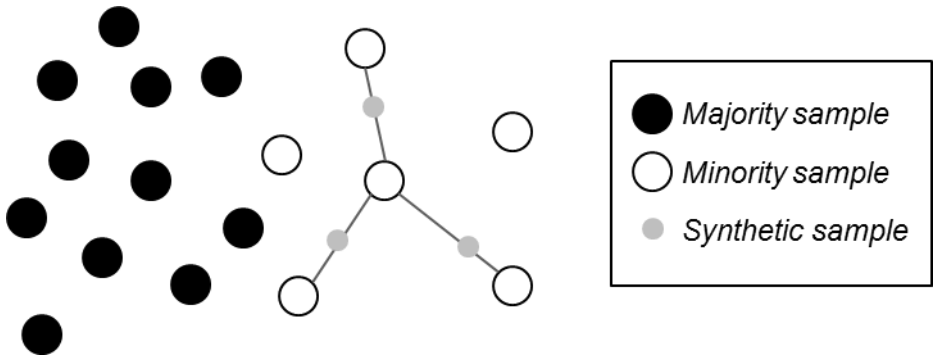

Figure 10: SMOTE Conceptual View

4.1.2 Unsupervised Learning

In unsupervised learning, the task is to find a structure in unlabeled samples. Unlike supervised learning in which the model is trained using inputs with known target output, in unsupervised learning the model seeks to predict the outputs of an unlabeled input without prior knowledge of its target output. This section introduces clustering, and in particular K-means clustering, which is a common approach for unsupervised learning and the technique leveraged in our experiments.

4.1.2.1 Clustering

The general idea behind clustering is that samples can be grouped into separate clusters based on some similarity measure based solely on the information and relationships found in the samples. Samples in one cluster tend to be more similar to one another and different from other samples in another cluster. The greater the similarity within a group, the greater the difference between groups leading to more defined clusters [44, p. 490].

Clustering can be categorized based on the final structure of the clusters. For example, a common differentiation among clustering types is whether the clusters are hierarchical or partitional. When clusters have sub-clusters, it is said to be a hierarchal clusters in which each cluster is the union of its sub-clusters. On the other hand, a partitional clustering, also referred to as exclusive clustering, denotes that each data point is a member of exactly one cluster (non-overlapping clusters). In cases where a data point can be assigned to two clusters (halfway between two clusters), it results in an overlapping clustering type. In situations where a data point cannot be assigned to any well-defined clusters, it is left out and consequently the clustering is said to be partial when compared to a complete clustering in which every data point belongs to a cluster [44, pp. 491–493].

4.1.2.1.1 K-means

K-means is a technique for producing clusters based on a prototype, which is defined in terms of a centroid. As the name indicates, the centroid in k-means is the mean

of a collection of data points. Each data point is closer to the center of its cluster than the center of any other clusters. Therefore, k-means is said to produce a center-based, or a prototype-based, clustering. K-means can be explained in the following protocol:

1) Select *K* points as initial centroids
2) ∀ data point, assign the data point to its closets *K* centroid
3) ∀ cluster, recalculate the centroid
    3.1) If centroid changes, repeat step 2
    3.2) If centroid does not change, stop

Figure 11 demonstrates how k-means works iteratively on defining clusters.

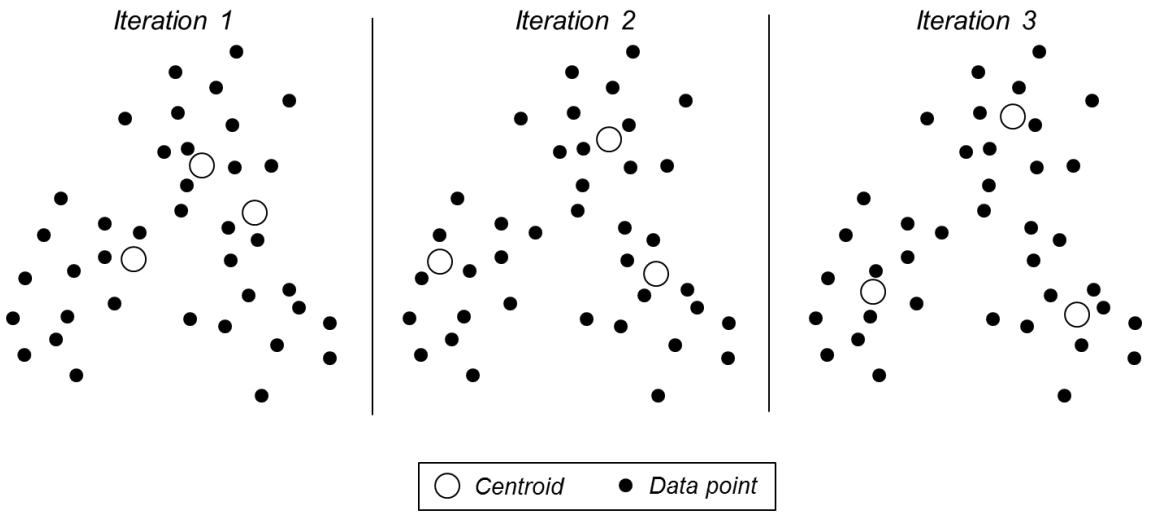

Figure 11: Creating clusters using k-means algorithm

4.1.3 Methods Reference

Table 1 provides a mapping between the machine learning methods used and the experiments in this research.

Table 1 - Mapping of Applied Machine Learning Methods to Thesis Sections

Method	GUI-based Application Command Streams	Scrolling Behavior based Active Authentication
Decision Trees (C4.5)	6.4	
Naïve Bayes	6.4	
Random Forests	6.4	7.4.1
AdaBoost	6.4	7.4.3
SMOTE		7.4.2
K-means		7.4.5

CHAPTER 5: PERFORMANCE EVALUATION

This chapter describes the various metrics used in evaluating the performance of the proposed models and the general experiment design applied.

5.1 Metrics

For classification algorithms, several metrics ("performance indices") are used to evaluate the "goodness" of a classifier. The goal is to measure how well the classifier is able to predict the correct class of the object under scrutiny. These matrices are based on a common 2x2 table known as the confusion matrix. The following subsection describes the confusion matrix and the matrices that can be produced from it.

5.1.1 Confusion Matrix

A confusion matrix, also known as a contingency table, is a representation of the classifier's output. It helps in visualizing how an algorithm classified instances by assigning them to four classes (see Table 2):

- True Positive (TP): Predicted instances that were correctly classified belonging to a class
- False Positive (FP): Predicted instances that were incorrectly classified belonging to a class

- False Negative (FN): Predicted instances that were incorrectly classified NOT belonging to a class
- True Negative (TN): Predicted instances that were correctly classified NOT belonging to a class

Table 2: Confusion Matrix

		Predicted class	
		Positive Class	Negative Class
Actual class	Positive Class	**True Postive**	**False Negative**
	Negative Class	**False Postive**	**True Negative**

From the confusion matrix, several performance matrices can be derived including accuracy, precision, sensitivity, specificity and the F-measure.

5.1.1.1 Accuracy

Accuracy is the measure of how well a classifier is able to correctly assign an object to its class. It is the ratio of true prediction – both true positives and true negatives – to the total number of prediction and can be calculated using equation 5.1 as follows:

$$Accuracy = \frac{\sum TP + \sum TN}{\sum TP + TN + FP + FN} \qquad (5.1)$$

5.1.1.2 Precision

Precision is a measure of the classifier's accuracy provided that a specific class has been predicted. It is the ratio of true positive prediction to the total number of positive

prediction both true positives and false positives and can be calculated using equation 5.2 as follows:

$$Precision = \frac{\sum TP}{\sum TP + \sum FP} \qquad (5.2)$$

5.1.1.3 Sensitivity (Recall)

Sensitivity, also known as recall or the true positive rate, is the measure of how well the classifier is able to assign instances to the positive class. It is the ratio of true positive predictions to the total number of instances belonging to the positive class and can be calculated using equation 5.3 as follows:

$$Sensitivity = \frac{\sum TP}{\sum TP + \sum FN} \qquad (5.3)$$

5.1.1.4 Specificity

Specificity, also known as the true negative rate, is the measure of how well the classifier is able to select instances to the negative class. It is the ratio of true negative predictions to the total number of instances belonging to the negative class and can be calculated using equation 5.4 as follows:

$$Specificity = \frac{\sum TN}{\sum TN + \sum FP} \qquad (5.4)$$

5.1.1.5 F_1-measure

The F_1-measure, also known as the F_1-score, is another measure of the classifier's accuracy. It is the harmonic mean of precision (see section 5.1.1.2) and recall (see section 5.1.1.3). The F_1-measure is a preferred performance matrix when working with imbalanced datasets especially when the minority class is more important. This is due to

the fact that when calculating the F_1-measure for each class, one can see how well the classifier was able to predict the minority class. Such observation is not clear when evaluating the classifier on precision or recall independently. The F_1-measure can be calculated using equation 5.5 as follows:

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \qquad (5.5)$$

5.1.2 ROC Curve

The Receiver Operating Characteristic (ROC) curve is a graphical plot that helps visualize the performance of a classifier by illustrating the tradeoff between sensitivity and specificity. It is generated by plotting the true positive rate against the false positive rate on the vertical and horizontal axes respectively. Figure 12 helps illustrate this concept further. The closer the curve to the top-left corner, more accurate the prediction power of a classifier. On the other hand, the closer the curve to the bottom-right corner, less accurate the prediction power of a classifier. The closer the curve to the straight linear line that splits the ROC space into two equal halves (from the bottom-left corner to the top-right corner) indicates that the classifier's prediction power is random. Figure 12 also illustrates how one can visually compare the performances of two classifiers. Classifier 1 (illustrated as a dash-dot line) is more accurate than classifier 2 (illustrated as a dash-dot-dot line).

The Area Under the ROC Curve (AUC) represents the probability that a randomly selected positive class instance is assigned a higher score than a randomly selected negative class instance. The value of the AUC ranges from zero to one where zero

indicates that the classifier has a false positive rate of 100% with a true positive rate of 0%; and one indicates that the classifier has a false positive rate of 0% with a true positive rate of 100%.

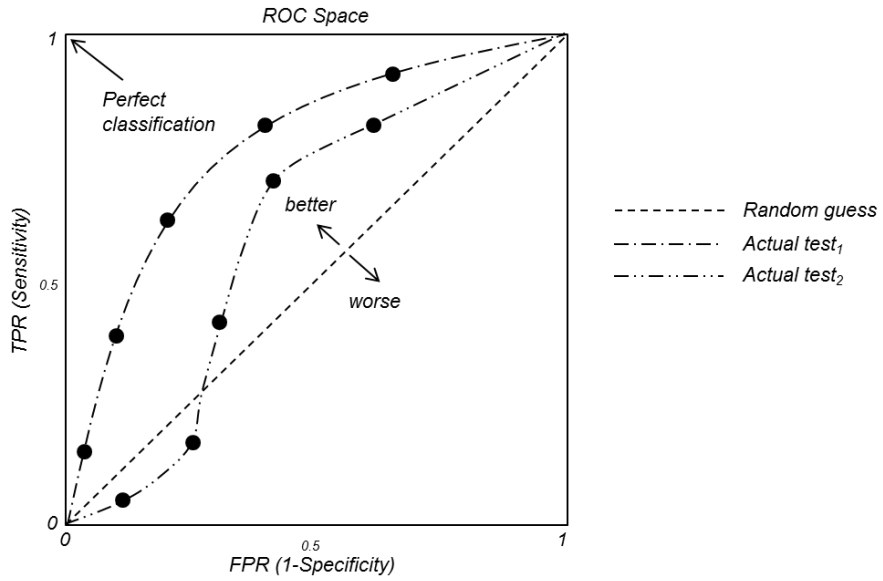

Figure 12: ROC Space

5.2 Experimental Design

Experimental design is structured around a set of key principles including the objective of the experiment, data representation, performance validation and performance evaluation. This section describes how each of these concepts is applied in our experiment design.

5.2.1 Objectives

Experimental design begins with identifying the objective of the work. In our case, the ultimate goal is to create a novel authentication model that can actively test for user authenticity based on the user's interaction with a given document. This can be further broken down into three specific tasks:

1. Find if a unique pattern exists to how a user interacts with the document. We explore user/document interaction in terms of the command the user issues from a GUI application and the way a user scrolls over a document.
2. Use this unique pattern to build a user profile that will be used for identification or verification of a user in an active authentication model.
3. Determine a machine learning algorithm that is capable of matching this profile to an observed user behavior with acceptable rate of accuracy.

5.2.2 Data Representation

Data representation refers to the features selected and/or extracted from the dataset used to describe, or represent, an object. Collectively, these features form a feature vector. One can think of an object as a dot in an N-dimensional feature space. Constructing a useful feature vector is an important aspect of experimental design since the dataset may contain many ineffective features such as redundant and irrelevant ones. Redundant features are those features that do not add any new information to the feature vector. For example, both a social security number and a student ID number may serve the same purpose, which is a unique identifier of an object. In this case, one can use one

or the other to provide the same information. On the other hand, irrelevant features are those that do not provide useful information within context to the feature vector. For example, if the goal is to construct a feature vector that represents a student's academic performance, the student's height may be irrelevant (unless the goal is to find if there is a relation between a student's height and their academic performance – most likely not).

To construct a feature vector, one can select the desired features or and/or extract additional ones. Extracted features are derived functions from the original features. For example, extracting a feature that represents the cumulative GPA for a student. Data representation is important as the goal is to select and/or extract those features that are best at discriminating an object.

In our research, feature selection was done through a trail-and-error approach in which features were added and removed creating multiple feature vectors to determine those that improve generalization and interpretability of the classifiers used. Detailed discussion of the specific data representation is provided in sections 5.2.2 and 6.2 where we describe the data representation of each active authentication model.

5.2.3 Performance Validation

Another aspect of experimental design is how to validate the performance of the selected algorithm. Since performance in measured by the true error rate the goal is to select the algorithm that provides the lowest error rate on the entire population. However, since practically we do not have access to an infinite number of samples, the algorithm must be trained and tested with the available dataset.

A common approach is to split the dataset into two subsets: a training set used for training the model; and a test set used for testing the performance of the trained model. This approach is often referred to as the holdout method and a 20/80 split is a common application where 20% of the dataset samples are used for training and the remaining 80% is used for testing. However, this method has its drawbacks. For example, if the dataset size is small, splitting the dataset may result in even smaller subset to be considered statistically significant. Moreover, the split may create bias distribution of the samples where similar samples are overwhelmingly belonging to one subset, hence resulting in a misleading error rate.

To overcome the limitations of the holdout method, a set of resampling techniques can be applied including random subsampling, leave-one-out cross validation and k-cross validation. In random subsampling, each split randomly selects a fixed (without replacement) number of samples and the model is retrained using the training samples and tested using the test samples. However, since the method selects a fixed sample without replacing them, not all samples are used for training and testing the model.

Leave-one-out cross validation is an exhaustive cross validation method in which learning and testing is performed on all possible ways by dividing the dataset into training and testing sets. For N samples, the method will run N trials where each trial used N-1 samples for training and the remaining samples for testing. This technique is best suited when the sample is small due to costly computational time with large number of samples. In our experiments, this method is used with k-means clustering as described in section 7.3.2 in this paper.

When dealing with a large number of samples non-exhaustive cross validation methods are preferred. Such methods include K-fold cross validation, which is similar to the random subsampling method with the exception that all samples in the dataset are used for training and testing the model. In a k-fold cross validation the dataset is stratified into K distinct folds consisting of 1/k of the total data. The model is then training using k-1 folds and the remaining one for testing. This is repeated until all folds are used for both training and testing. In our experiments, k-fold cross validation was applied in validating our classification method as described in section 7.3.1.

Selecting the ideal numbers of folds is not a straightforward task and depends on the size of samples on hand. However, one should consider the following tradeoff. The larger the number of folds is, the smaller the bias of the true error rate will be but its variance will be high. On the other hand, the smaller the number of folds is, the larger the bias of the true error rate will be, but its variance will be small. Therefore, larger number of folds is more suited in situations where the dataset is large. However, with sparse dataset, a method such as leave-one-out is preferred so that the algorithm can be trained on as many samples as possible. In our experiments, in cases where k-folds cross validation was applied we chose the common practice of assigning k = 10.

5.2.4 Performance Measurement

Deciding on the performance metrics is another important aspect of experimental design. As discussed in CHAPTER 5: Performance Evaluation, different metrics can be used to measure the performance of a model. While all of them are considered, we assign

a higher weight on the F_1-measure due to it being insensitive to the true negative rate, which represents the majority class in our datasets (the minority class is the class of interest to us). By examining the F_1-measure for each class, we can determine how well the algorithm was able to predict each class.

CHAPTER 6: APPLICATION COMMANDS STREAMS AUTHENTICATION MODEL (ACSAM)

This section presents and describes an active authentication model based on behavioral biometrics pertaining to GUI-based application user-issued commands. Researchers have explored the idea of building users profiles based on users' behavioral patterns when interacting with such graphical interfaces, especially in the area of human-computer interaction. They did so by analyzing the user's keystroke and/or mouse dynamics. However, most of the work that focused on creating behavioral model from user issued commands, and perhaps the most relevant to this work, were limited to users' interaction with the *Nix command shell program [39]–[42]. Never before the idea of creating profiles by capturing users' usage characteristics when interacting with a specific application's GUI has been examined, this goes beyond how a user strikes the keyboard or moves the mouse across the screen. It provides more dimensions to consider and a richer set of behavioral features to include when building behavioral profiles.

This model relies on the user's issued commands when interacting with a GUI-based application. It collects data points (e.g., commands) from the user's interaction with the application (e.g., MS Word), builds a user profile and monitors further user interactions to make a determination of the user's authenticity. To answer the question of whether or not a stream of commands triggered by users' interaction with a GUI-based application

can serve as a behavioral biometric, we have chosen to utilize and repurpose a dataset collected by the MITRE Corporation from previous research on organization-wide learning and recommender systems [46]. This dataset represent two years of users' usage data from interacting with Microsoft Word (see section 6.1). The motivation behind our consideration of the MITRE dataset is due to the following:

- MS Word is a more popular medium among the common computer user than the *Nix command shell program. Microsoft claims that roughly half a billion people use MS Office [47]. Furthermore, a recent report by Forrester shows that 84% of organizations are using MS Office 2010 as their office productivity suite [48]. Therefore, creating user behavioral profiles from interacting with such widely used applications would have a broader applicability.
- The graphical user interface of MS Word provides a richer set of behavioral features when compared to the command line interface (CLI) of the Linux shell. The possible feature set is unique to the application in use and can be derived from the way a user interacts with its properties. While the current dataset is strictly a representation of the commands issue by users, a future consideration would explore a richer set of behavioral features such as the average velocity of scrolls, the ratio of backward and forward scrolling switches and the average elapsed time between scrolls, to list a few.
- The last, and most obvious, reason is the availability of the dataset from a previous work by MITRE that is made public.

The following subsections describe the dataset, data representation, experiment design and performance evaluation of the proposed active authentication model.

6.1 Dataset

The data used for this model was collected by the MITRE Corporation over a period of two years from 1997 to 1998. MITRE monitored commands entered in Microsoft Word version 97 by 24 employees. The employees consisted of artificial intelligence researchers as well as technical and support staff. All individuals used the Macintosh operating system. As the system became more robust, additional users were monitored. If an individual switched to a PC, they were dropped from observation. The data was initially collected to explore the development of recommender systems that leverage knowledge of how the entire group used Microsoft Word to tailor the application to the organization [13]. The dataset is made publicly available at http://www.research.rutgers.edu/~sofmac/ml4um/. A sample of this dataset is provided in

Appendix B: Sample Data From the MITRE dataset.

Each time a user opened Microsoft Word and executed a command during the period of observation the command was logged along with a unique user ID, the version of Word they used, the file size, the creation date of the file, the operating system and version and the date/time the command was executed. The commands represent editing actions executed inside Microsoft Word such as "*Copy*", "*Paste*", "*New*", or "*Italic*". Over the course of the study, a total of 74,783 commands were observed. The users executed 174 unique commands, and on average, each user executed 1,583 commands. Users who participated in less than 10 sessions where discarded from the data analysis.

6.2 Data Representation

The MITRE dataset was repurposed to answer the question of whether or not there exists a unique pattern in how a user edits a document inside a GUI-based application, and therefore the executed stream of commands can serve as a behavioral biometric. We treat this as a binary classification problem with the goal of classifying a particular document editing behavior as authentic (belonging to the authenticated user) or not (belonging to someone else).

To do this, a bag-of-words model is used for classifying a new document editing behavior, where the number of occurrence of each command is used as a feature for training the classifier. A profile is created for each user consisting of observed number of commands executed consecutively on a single document. A working session is defined

by the series of commands executed by a particular user and identified by the captured user ID. Because file names are not captured in the data, the commands were organized into sessions using a composite key of the *file creation date* and *file size*. It is observed that in the dataset the file size for a document remains constant as long as the document remains open. All commands belong to a single session as long as no more than 60 minutes elapse between issued commands.

Consequently, the 74,783 commands coalesce to 11,334 sessions; an average of 6.59 commands per session. The longest and shortest session consisted of 93 and 1 commands respectively. Three of the 24 subjects participated in fewer than ten sessions. Data for these users are discarded. There is an average of 539 sessions per user. After organizing the data by reading sessions, all sessions belonging to an individual contribute to generate that user's command profile. Table 3 summarizes the statistics above.

Table 3: Aggregate summary of the MITRE dataset

Data	Value
Total number of issued commands	74,783.00
Total number of unique commands	174.00
Average number of commands per user	1,583.00
Total number of sessions	11,334.00
Average number of commands per session	6.57
Average number of sessions per user	539.00
Longest session by number of commands	93.00
Shortest session by number of commands	1.00

6.3 Experimental Design

The goal of the experiment is to determine whether knowledge of a user's profile (the authenticated user), as well as knowledge from other users' profiles (imposters), is sufficiently distinctive to allow for differentiation between the authenticated user and masqueraders. This question is a natural fit for machine learning. To answer this question, we compared the performance of commonly used machine learning algorithms with the following protocol:

For each user u:

1) Label sessions belonging to user ***u*** as *authenticated*
2) Label sessions not belonging to ***u*** as *imposter*.
3) Perform a 10-fold cross validation:
    3.1) For each fold ***f***:
        3.1.1) Train classifier on all folds $\neq \boldsymbol{f}$.
        3.1.2) Label each session in ***f*** as *authenticated* or *imposter*.

In a 10-fold cross validation, the sessions are stratified into ten distinct folds consisting of $^1/_{10}$ of the total data. Class distribution (the ratio of authenticated vs. unauthenticated sessions) is preserved in each fold. A classifier algorithm is trained on 9 of the 10 folds. The classifier is asked to label each session in the test fold as *authenticated* or *imposter*. This process is repeated 10 times, with each fold being used as test data once.

Each user is treated as the authenticated user once. Classifier performance is averaged across users. We evaluated several different well-known classifier algorithms including C4.5 decision tree, Naïve Bayes, Adaptive Boosting (AdaBoost), and Random

Forest. As a baseline, we used a majority class classifier that always classified the instance as "*imposter*". The following metrics were used to compare classifier performance: percentage of correctly classified sessions, F-measure, AUC of the Receiver Operator Characteristic (ROC) curve. The larger those metrics are the better performance is.

In addition, due to the imbalanced nature of the dataset, we have observed that the class distribution is extremely skewed. The number of sessions labeled "*imposter*" vastly outnumbers the number of sessions belonging to the authenticated class. We examined changes in classifier performance when sessions labeled "*imposter*" are subsampled to 10% of their original number in the training data. Similarly, we used a 10-fold cross validation for each user's profile as after subsampling.

6.4 Performance Evaluation

We first compare the ability of the selected classifier algorithms to distinguish between the *authenticated* and *imposter* users (see Table 4). The evaluation procedure by which each user profile is once treated as the *authenticated* user and then repeatedly as *imposter* is described in detail in the previous section. The Random Forest algorithm with 20 decision trees was trained on a randomly selected subset of eight features. AdaBoost was trained in 10 iterations with a decision stump as the base classifier.

Table 4: Average algorithm performance across user profile using 10-fold cross validation

Algorithm	Average percent correct	Average F-measure	Average AUC

Random Forests	95.43%	0.943	0.735
AdaBoost	95.07%	0.931	0.674
Naïve Bayes	95.01%	0.932	0.595
C4.5	95.42%	0.939	0.566
Baseline	95.04%	0.927	0.500

All classifiers outperformed the baseline in terms of the F_1-measure and AUC, if only narrowly in some cases. Random Forest bested the other methods in terms of all three metrics. In particular, AUC was much higher for Random Forest than the other algorithms tested. Although the perceived increase in average percent correct is moderate for Random Forest, a comparison of a confusion matrix (see Table 5 and Table 6) between Random Forests and AdaBoost, the second best algorithm, reveals that Random Forest is identifying the minority/rare class (*authenticated*) at a better rate.

Table 5: Random Forests confusion matrix

		Predicted Class	
		Authenticated	*imposter*
Actual Class	*Authenticated*	10.59	45.66
	imposter	6.19	1072.57

Table 6: AdaBoost confusion matrix

		Predicted Class	
		Authenticated	*imposter*
Actual Class	*Authenticated*	3.49	52.79
	imposter	3.29	1079.50

Random Forest and AdaBoost both perform well and both are ensemble learning methods. Overall, Random Forests performs better due to its intrinsic characteristics (e.g. randomness and sampling/bagging), and ensemble methods, which cope better with the varying nature of individual user profiles. This level of optimal performance is in line with other research [5], [49], [50], [51] which has found that ensemble learning, either through boosting or bootstraps aggregating (bagging) are flexible enough to represent complex hypothesis functions that are difficult to learn with a single classifier. In addition, bagging (used by Random Forest) can reduce the concern that the learned model may over fit the dataset.

While these initial results suggest that it is possible to use application-specific command streams to distinguish between authenticated users and others in Microsoft Word, the rate at which authenticated users are misclassified as imposters (a false negative) could be prohibitively high in practice (this would depend on the application and the action required when an active authentication system triggers a false negative).

To reduce the false negative rate, the majority class was subsampled in the training set to 10% of its total size, bringing the number of sessions in the *authenticated* and *imposter* classes much closer to an even split. Results for this experiment are presented in Table 7.

Table 7: Average algorithm performance across user profiles with majority class subsampled to 10%

Algorithm	Average percent correct	Average F-measure	Average AUC
Random Forests	85.76%	0.879	0.746
AdaBoost	87.56%	0.876	0.675
Naïve Bayes	58.54%	0.614	0.649
C4.5	85.12%	0.871	0.668

The results show that subsampling the majority class makes all the classifiers more sensitive to the minority class. Subsampling is only used in the training set, not the test set, as this would inaccurately represent the observed class distribution. Although the average percent correct rate declines by 9.67% for the Random Forest algorithm, the average AUC increases. This is an improvement to the classifier's performance since the better the classifier performs, the higher the AUC [52]. The relative significance of AUC comes from the derivation and interpretation of ROC where decision-making is a function of setting thresholds on similarity distances to trace the ROC curve.

Table 8 presents the confusion matrix for the Random Forest algorithm trained on the subsampled training data. In comparison to the initial confusion matrix for the Random Forest algorithm (see Table 5) trained on the original training data, subsampling has made the classifier far more sensitive to classifying authenticated sessions as *authenticated* (true positives have increased). Subsampling the majority class thus results in a trade-off: a substantially lower false negative comes at the cost of potentially confusing additional *imposter* sessions as *authenticated* (an increase in false positives).

Table 8: Random Forests confusion matrix after subsampling the majority class

		Predicted Class	
		Authenticated	*imposter*
Actual Class	*Authenticated*	36.96	17.01
	imposter	144.06	936.47

Table 9 presents a comparison between ACSAM and other related research in terms of novelty, accuracy and significance.

Table 9: ACSAM vs. Others

Model	**Novelty**	**Accuracy**		**Significance**	
		Research by	Results	# Users	# Commands Issued
ACSAM	**GUI-based Application Command Streams** Based on number of command (bag-of-words) issued by user from a GUI-based application	El-Masri et al. (2014)	95.43%	24	74,783
Others	**Command-line lexicon** Similarity measure of sequence of fixed-length command	Marin et al. (2001)	80.00%	50	5,000
	Percentage of commands issued in a given time period	Yeung & Ding (2003)	87.00%	08	2,356
		Schonlau et al. (2001)	80.40%	70	15,000
	Frequency of commands issued	Maxion & Townsend (2002)	74.15%	50	15,000

CHAPTER 7: SCROLLING BEHAVIOR BASED AUTHENTICATION MODEL (SBAM)

We describe here a novel behavioral biometric based on users' document scrolling traits, and then introduce a new method that leverages this unique trait for re-authenticating users. It is important to note that scrolling occurs as the result of using a mouse and/or the keyboard keys. In particular, we focus on identifying anomalous scrolling behavior when users interact with protected or read-only electronic documents. This poses a unique challenge due to the minimal user input that can be observed and analyzed for authenticity. Protected documents such as protected Microsoft (MS) Word files and Portable Document Format (PDF) files prohibit input thereby facilitating our focus on observing activities beyond the traditional modification of, addition to and/or deletion of documents' contents. Hence, this method relies only on how those documents are viewed.

The goal of this experiment is to investigate the existence of patterns related to how users scroll electronic documents. Such patterns are influenced by many factors including the users' purpose or intent; the documents' contents, length, layout, and type; the environment in which such interaction takes place; users' knowledge of the subject; rate of interruptions; day of week and time of day; the platform used to view documents and users' physical condition. While some of these factors can be controlled such as

environment and document layout, others are difficult or impossible to control such as users' physical condition. Therefore, the challenge is to find out if a unique reading pattern exists regardless of external factors.

7.1 Dataset

We have repurposed a dataset that was obtained from a previous research project at the Georgia Institute of Technology (Georgia Tech). This dataset was culled from an experiment aimed to detect document access activities that indicate evasive cyber insider attacks such as those that use various strategies to escape generic thresholds on coarse-grain metrics (e.g., indiscriminate bulk read or bulk copy of documents) [53]. The experiment required subjects to login into a custom-built web application and read a loaded PDF document within a browser. The users' reading habits were tracked throughout the reading session and subsequently logged to a database. Eighty-four subjects participated in the experiment and fifty-four documents were available for reading. All of the documents were either academic research papers or journal entries.

The web application was programmed to record the browser's *onResize* and *onScroll* events. The *onResize* event is triggered every time the browser's window is resized, while the *onScroll* event is triggered whenever the horizontal or vertical scroll bars' positions are changed. Every time the user resized the browser window or scrolled through the page, the following data points were captured and stored to a database: the vertical and horizontal scroll position; the document width and height, the window width and height, and the time the event took place. Table 10 provides descriptions for each of these data points.

Table 10: Data logged at each *onResize* and *onScroll* events

Data Point	Description
Horizontal scroll position	The X coordinate of the top-left-most visible pixel on the screen.
Vertical scroll position	The Y coordinate of the top-left-most visible pixel on the screen.
Window width	The width of the browser window in pixels.
Window height	The height of the browser window in pixels.
Document width	The width of the document in pixels. This changes when a user zooms in or out.
Document height	The height of the document in pixels. This changes when a user zooms in or out.
Time	The timestamp of the log entry in UNIX timestamp format (milliseconds elapsed since January 1, 1970).

The terminology aligned with the above web application goes as follows:

- Observation (O): An observation is a captured event of a user/document interaction.
- Working Session (WS): A working session is the record of one user's observed interactions with a single document during a single continuous period of interaction. A working session comprises one or many observations.

Different users read a different number of documents and collectively produced a total of 529 reading sessions.

7.2 Data Representation

Three feature vectors were derived and used with various classification and clustering techniques. The feature vectors were fed to machine learning algorithms to select features ("human factors") that discriminate best between different readers prior to classification. Description of each feature vector is presented in the subsequent sections.

7.2.1 Feature Vector I

Feature Vector I is a pure statistical representation derived from the user's scrolling traits. It is comprised, among others, of the average standard deviation and the average velocity of the user vertical and horizontal scrolling over the document. Fourteen features were extracted here. Table 11 provides a description of each of the extracted features.

Table 11: Feature Vector I data description

Data Point	Description
Number of logs	Total number of events logged for the reading session.
Average scroll Y distance	The average vertical distance in pixels between subsequent logged events.
Scroll Y distance standard deviation	The standard deviation of the vertical distance in pixels between subsequent logged events.
Average scroll X distance	The average horizontal distance in pixels between subsequent logged events.
Scroll X distance standard deviation	The standard deviation of the horizontal distance in pixels between subsequent logged events.
Average elapsed time	Average amount of time in milliseconds between two subsequent logs
Elapsed time standard deviation	Standard deviation of time in milliseconds between two subsequent logs
Average velocity	Average velocity in pixels per second between subsequent logged events.

Velocity standard deviation	Standard deviation of pixels per second velocity between subsequent logged events.
Switch / scroll ratio	Number of changes in scroll direction divided by number of logged events.
Number of switches	Number of times a user switched directions while scrolling.
Number forward scrolls	Number of events logged as a result of forward scrolls.
Number backward scrolls	Number of events logged as a result of backwards scrolls.
Total elapsed time	Total time that the document was read in milliseconds.

7.2.2 Feature Vector II

Feature Vector II is a representation of the polarity of scrolling (forward vs. backward scrolls) and the pauses between scrolls. Here we investigate the possibility that there exists a unique pattern in how a user scrolls backward and forward while reading a document. Each working session is divided into a series of 5-grams sequences. Each of the entries in the 5-gram corresponds to a scroll movement by the user. Through experimentation, we found that 5-gram length sequence provided sufficient information to evaluate.

For example, let *f* denote the user's forward scrolling, and let *b* denote the user's backward scrolling. If the user had moved forward 3 times, backwards one time and forward one time the corresponding 5-gram would be [*f, f, f, b, f*]. A working session consists of all 5-grams generated during the users' interaction with the document. The individual 5-grams for a working session are summarized into a set of 15 features that keep track of whether that 5-gram appeared more frequently than one would expect based

on all working sessions. Each feature is binarized to zero or one based on whether that feature appears more often than we'd expected (feature value = 1) or less often (feature value = 0). This representation is comprised of the 5-gram scroll sequence, the average and standard deviation of the sub-sequence distance and pauses. A total of 15 features were extracted for this feature vector. Table 12 provides a description of each of the extracted feature for this vector.

Table 12: Feature Vector II data description

Data Point	Description
Sequence	The current extracted sequence of 5-gram length.
User ID	The user identifier.
Total forward scroll distance	The total distance in pixels of the user's forward scrolling.
Average forward scroll distance	The average vertical distance in pixels between subsequent logged events when user scrolled forward through the document.
Average forward scroll velocity	Average velocity in pixels per second between subsequent logged events when user scrolled forward through the document.
Total backward scroll distance	The total distance in pixels of the user's backward scrolling.
Average backward scroll distance	The average vertical distance in pixels between subsequent logged events when user scrolled backward through the document.
Average backward scroll velocity	Average velocity in pixels per second between subsequent logged events when user scrolled backward through the document.
Total time	Total time that the document was read in milliseconds.
Average time	The average of time in milliseconds between pauses.
Time standard deviation	The standard deviation of time in milliseconds between pauses.
Total forward pause	Total number of pauses between subsequent forward scrolling.

Average forward pause	The average number of pauses between subsequent forward scrolling.
Total backward pause	Total number of pauses between subsequent backward scrolling.
Average backward pause	The average number of pauses between subsequent backward scrolling.

7.2.3 Feature Vector III

In this data representation, we treat the dataset as a bipartite graph with two disjoint node sets - ***U*** for users and ***D*** for documents - where undirected edges represent a read-by relation. This feature vector will produce the best results at capturing what is most discriminative between users and imposters during the same session. The notation here is as follows:

- Denote by $U = \{u_1, u_2, \ldots, u_n\}$ the set of all users where $|U| = 46$
- Denote by $D = \{d_1, d_2, \ldots, d_n\}$ the set of all documents, where $|D| = 65$
- Denote by $u_i \rightarrow d_j$ the relationship "user u_i works with document d_j"

The following criteria are applied to the data selection process:

- Exclude documents that have been worked on fewer than 5 times – $deg(d_i) >= 5 \quad \forall d_i \in D$
- Exclude users who have not worked on at least 8 documents – $deg(u_i) >= 8 \quad \forall u_i \in U$

Consequently, we are left with $|D| = 35$, $|U| = 27$ and $|E| = 292$ (the size of the edge set based on actual number of undirected edges or the $u_i \rightarrow d_j$ relations). Each edge

corresponds to a working session, which corresponds to a single user working on a single document. Edges are labeled with a feature vector extracted for that working session. Each working session is binarized using the following method: 1) Calculate median values for each of the features across all users and all documents. 2) Binarize all feature vectors based on whether the given value is above the median "1" or below the median "0". A total of 23 features were extracted in this feature vector. Table 13 provides a description of each of feature.

Table 13: Feature Vector III data description

Data Point	Description
Number of entries	The total number of events logged for the reading session.
Post ID	The logged event identifier.
Document ID	The document identifier.
User ID	The user identifier.
Pause average	The average amount of time in milliseconds between two subsequent logs.
Pause standard deviation	Standard deviation of time in milliseconds between two subsequent logs.
Number of pauses	The total number of pauses.
Pauses to length ratio	The total number of pauses to the total distance.
Number of long pauses	The number of pauses that lasted more than a specified threshold.
Long pauses to length ratio	The total number of long pauses to the total distance.
Distance average	The average of the vertical distance in pixels between subsequent logged events.
Distance absolute average	The absolute average of the vertical distance in pixels between subsequent logged events.

Distance standard deviation	The standard deviation of the vertical distance in pixels between subsequent logged events.
Distance absolute standard deviation	The absolute standard deviation of the vertical distance in pixels between subsequent logged events.
Number of forward switched	Number of events logged as a result of forward scrolls.
Number of backward switched	Number of events logged as a result of backward scrolls.
Number of switches	Number of times a user switched directions while scrolling.
Number of forward scrolls	Number of events logged as a result of forward scrolls.
Number of backward scrolls	Number of events logged as a result of backward scrolls
Switches to scrolls ratio	The total number of switches to the total number of scrolls.
Switched to length ratio	The total number of switches to the total distance.
Velocity average	Average velocity in pixels per second between subsequent logged events.
Velocity standard deviation	Standard deviation of pixels per second velocity between subsequent logged events

7.3 Experimental Design

This section describes the authentication methods used, the feature representation they have access to, and motivate linkages and use.

7.3.1 Classification

The question we have to answer is: "Is the current user the authenticated user or not?" and consequently the problem is posed as a binary classification problem with two classes: *Authenticated* and *Imposters*. As a first cut, one can use Random Forests, an ensemble learning method for classification. Compared to other learning methods used, Random Forests performs much better due to its intrinsic characteristics, e.g. random

sampling and ensemble methods, which cope better with the varying nature of individual user profiles. That is, Random Forests trains multiple "weak" classifiers, in our case Decision Trees, based on a randomly selected subset of the entire feature space. The algorithm then makes the classification decision based on the majority vote of these weak learners. Random Forests employs Feature Vector I.

7.3.1.1 Handling Imbalanced Data

Due to the imbalanced nature of our dataset, SMOTE and AdaBoost techniques were applied independently to decrease the classifier's bias towards the majority class (*Imposters*) and improve the classification performance on the minority class (*Authenticated*). Using Random Forests as the classifier, we oversampled the minority class using SMOTE, a widely used technique for oversampling imbalanced data by creating synthetic minority instances along the line segments connecting the minority class nearest neighbors [45]. To improve the performance of the classifiers, we considered AdaBoost, a widely applied cost-sensitive boosting technique that boosts the outcome of the "weak learner" by assigning weight or cost in favor of misclassified instance [11]. This has the potential for improving the performance of the classifier on our imbalanced data. To this extent, we applied AdaBoost with Decision Stump and ADTree as weak learners. Again, Feature Vector I was used.

Another technique we applied to reduce classifier bias is sub-sequencing. Here, instead of extracting a single feature vector from a single reading session, we built a continuous feature vector across all reading sessions for a given user. In this document-

agnostic approach, we considered extracting features from n-gram sequences of scroll events. For each n-gram, we extracted the feature set for that sequence and tested it against Feature Vector II. Through experimentation, we found that 5-gram length sequence provided sufficient information to evaluate. For example, if a user had 300 scroll events for a single reading session, then that allowed a total of 296 feature vectors of type II to be extracted from the reading session in comparison to the single feature vector we had extracted previously. Each subsequence was essentially an estimate of the likelihood of the authenticated user being the reader during that subsequence. Consequently, as a user read a document, every new scroll event gave us a new subsequence for analysis.

7.3.2 Clustering

With modest success trying to predict the minority class in our dataset using classification, we turned our attention from authenticating the user identity to narrowing down the possibility that the current reading pattern belongs to a small set of registered user profiles. A pattern that cannot be associated with any of these profiles would be considered new or foreign, and hence belonging to a possible imposter. To achieve this, we considered clustering as a possible approach. Basically, the goal is to identify (or narrow down possibilities of) who is reading the current document.

7.3.2.1 Creating User Profiles

Leveraging Feature Vector III (see section 7.2.3 Feature Vector III) we extract a profile. To do so, each user's reading sessions are clustered using k-means and the

centroids are extracted from the resulting clusters. To determine the number of clusters we first used the simple and widely adopted rule of thumb, which sets the number of clusters to approximately $\sqrt{\frac{n}{2}}$ where *n* is the number of data points. Therefore, we probed for the number of clusters between one and $\sqrt{\frac{|RS_u|}{2}}$ where *u* is user. The number of clusters varied per user profile with the square root as the upper bound on the number of clusters. However, for a few profiles, we discovered that the squared error was still unacceptably high when we stopped introducing new clusters at $\sqrt{\frac{|RS_u|}{2}}$. As a result, we continued to add clusters until the squared error was less than 15 at which a balance between the intra-cluster variance and overfitting was achieved. A user's profile, therefore, could consist of one or many clusters.

7.3.2.2 Calculating Distance to Profile

Now that each user's profile can be represented as a set of centroids, we can attempt to determine who is reading a document based on a new reading session. Given a target reading session (an unlabeled reading session), user profiles are constructed from all reading session except the target (a Leave-one-out model). Then the distance from each profile to the target reading session is calculated. If there are *n* centroids in a profile, the distance to the profile is calculated as the minimum Euclidean Distance (ED) between the target and the cluster's centroids in the profile. This can be formulated as

$$dist(target, profile) = min_c \in centroids\ ED(target, centroid)$$

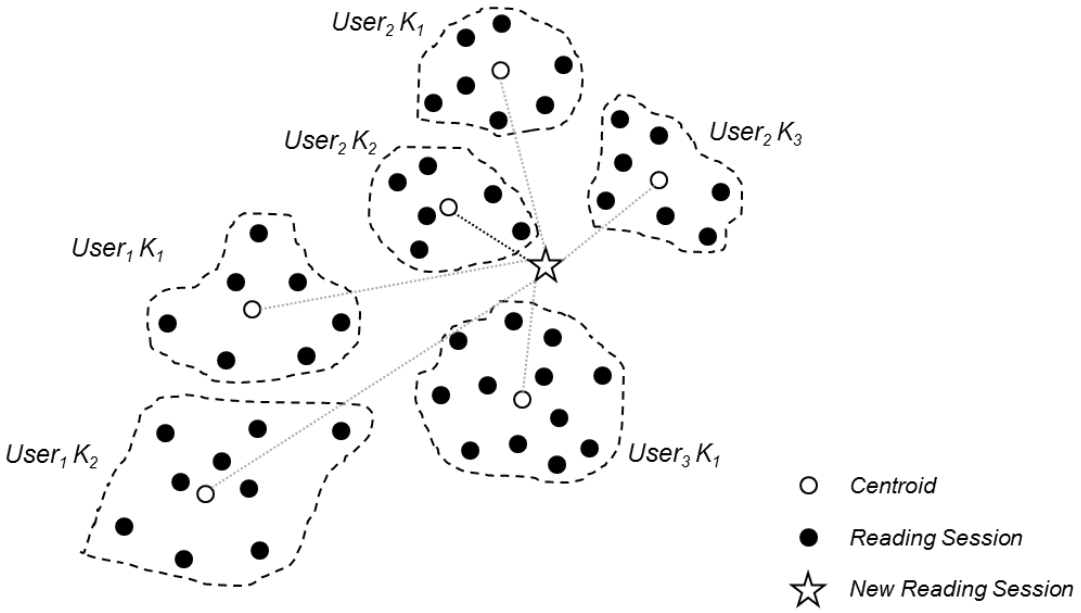

Figure 13: Users' Profile represented as K clusters

7.3.2.3 Determining Most Likely Readers

Now that the distance from the reading session to each profile is known, a set of most likely readers can be determined based on various metrics. Two approaches are followed and presented in the subsequent sections.

7.3.2.3.1 Approach 1: Top 5/10 Ranking (Singleton Sets)

The simplest approach to determining likely readers can be extracted by ranking the distances from a given working session to the profiles. This is repeated for all working sessions and profiles and recalculated with target working session removed.

7.3.2.3.2 Approach 2: Feasibility Sets (Multiple ID Sets)

Another approach to determining which users may have generated a working session is to label all profiles as possible candidates if target working session is within a threshold distance from that profile. A per-profile distance threshold is calculated based on the average distance and standard deviation between the different working sessions in each profile and the centroids. This gives a sense of how disperse each profile is. For each user we set the $threshold_u$ to the average distance plus one standard deviation from that average distance, hence expanding the notion of a cluster by adding a threshold ("leeway"). Our choice to add a single standard deviation is to reduce the risk of having the working session fall too close to other users' profiles. This allowed outliers to have better chance of being associated with the correct user since clusters could overlap allowing a new working session to fall within several profiles.

To determine the set of possible users, we choose a particular reading session as the target. The goal is to determine which profiles were capable of generating the target according to the distance from the profile's centroids to the target reading session. Of course, one profile clearly generated this reading session (the target was not included in the generation of the profile). A perfect system would mark only the actual user's profile as possible for each target. We mark each $profile_i \ \forall \ i \in U$ as possible or not possible. $Profile_i$ is marked possible if $dist(target, profile_i) \leq threshold_i$. Figure 14 illustrates the notion of expanding a cluster's boundary of a user's threshold.

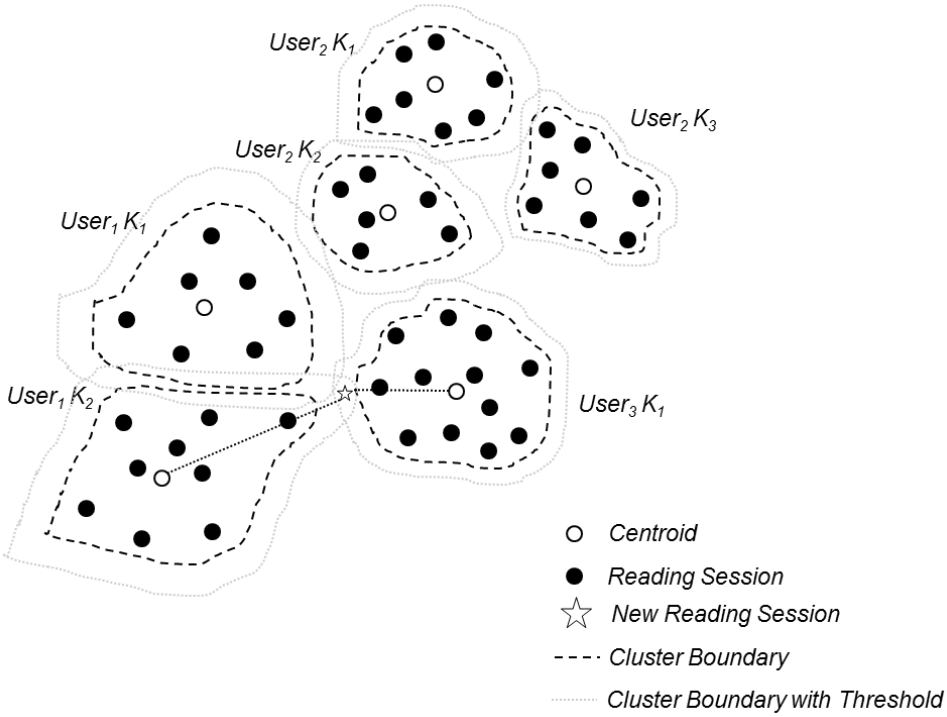

Figure 14: Expanding the notion of a k cluster by a threshold

7.4 Performance Evaluation

Due to the imbalanced nature of our dataset (the two *Authenticated* and *Imposter* classes are not approximately equally represented), we use the F_1-measure and AUC as performance measures. Relying only on the correct classification rate is misleading since classifiers tend to be bias towards the majority class (*imposter*). All classification experiments' performance involve k-fold cross validation and are summarized in Table 14.

7.4.1 **Random Forests**

For the purposes of our experiment, we eliminated sessions with less than 150 observations and users for whom we had fewer than six reading sessions. This left us with a total of 29 users. Random Forests with 10-fold cross validation was run on the stratified dataset represented by Feature Vector I (see section 7.2.1 Feature Vector I for details). Best results were obtained with 40 trees and 9 features achieving a modest F_1-measure of 0.27 for the Authenticated class (and 0.99 for the Imposter class.)

Moreover, tuning the classifier did not yield significant gain in performance. For example, increasing the number of features did not impact the model's accuracy. However, the model's error decreased as the number of trees increased. This observation plateaued after 40 trees. To provide the method with more descriptive data, we then only considered users that had at least seven reading sessions where each reading session lasted for a minimum of 30 seconds. This left us with data for 27 users. Again, the method performed poorly in predicting the minority class with an F_1-measure of 0.27 on the *Authenticated* class and a 0.98 on the *Imposter* class. Although discouraging, the results came as no surprise, as similar to many other practical classification problems, our dataset was imbalanced, thereby biasing the classifiers towards the majority class (*Imposters*).

7.4.2 **Random Forests with SMOTE**

To overcome the class imbalance problem, we applied SMOTE in three different configurations (see description of 3, 4 and 5 in Table 14). However, applying SMOTE

yielded modest improvement to the classifier's prediction power of the minority class with an F_1-measure of 0.29 (a slim 0.1 increase in performance), which is the best score of the three different configurations.

7.4.3 **AdaBoost**

Applying cost-sensitive boosting also resulted in a slight improvement in predicting the minority class. Here, AdaBoost with Decision Stump and again with ADTree was used achieving an F_1-measure of 0.35 and 0.39 respectively.

7.4.4 **Random Forests with Sub-Sampling**

In this experiment, we applied Random Forests with 10-fold cross validation. This resulted in an improved performance of the model that could be attributed to the larger number of cases tested. For example, in our previous experiment with AdaBoost the total number of cases, or predictions, for a given user was 500 compared to 503,662 predictions when applying sub-sequencing, and hence gaining a normal distribution over the produced instances. A total of 44 users were considered during this analysis and the performance has increased to an F_1-measure of 0.50 for the minority class (*Authenticated*), the model remained neutral in its prediction capability.

Table 14: Performance summary of classification experiments

#	Experiment	% Correctly Classified	% Incorrectly Classified	Authenticated class F-Measure	Imposter class F-Measure	AUC
1	Random Forests (Analysis run on all users that have > 5 *WS* AND there are > 150 *O* logged for each *WS*)	97.00%	3.00%	0.28	0.99	0.81
2	Random Forests (Analysis run on all users that have > 6 *WS* AND there are > 150 *O* logged for each *WS* AND *WS* is > 30 seconds) *	97.00%	3.00%	0.27	0.98	0.79
3	Same as experiment 2 but with 100% synthetic instances (SMOTE)	96.94%	3.06%	0.20	0.98	0.76
4	Same as experiment 3 but restricted to users who, after the iterative selection criteria in experiment 2 is performed, still have > 6 *WS*	95.97%	4.08%	0.29	0.98	0.82
5	Same as experiment 4 but with 500% synthetic instances (SMOTE)	95.75%	4.25%	0.26	0.98	0.78
6	AdaBoost with Decision Stump	97.92%	2.08%	0.35	0.99	0.88
7	AdaBoost with ADTree	97.94%	2.06%	0.39	0.99	0.80
8	Random Forests with Sub-sequencing	98.24%	1.77%	0.50	0.99	0.86

* Due to the iterative selection process, some users may end up with as few as 3 readings being analyzed

7.4.5 K-Means Clustering

With modest success trying to predict the minority class in our dataset using classification, we turned our attention from verifying the exact user identity to narrowing down the possibility that the current reading pattern belongs to a small set of registered user profiles. As described in sections 7.3.2.3.1 and 7.3.2.3.2, two approaches were followed: Approach I - Simple ranking by distance (Top 5/10 Ranking) and Approach II - filtering users by profile standard error (Feasibility Sets). Figure 15 summarizes the

results of approach I. It shows the cumulative percentage of the actual user was ranked against each position. Over all reading sessions, 19.75% of the time the actual user's profile is the closest to target, while 58% and 80% of the time the actual user was in the top five and ten ranked profiles respectively.

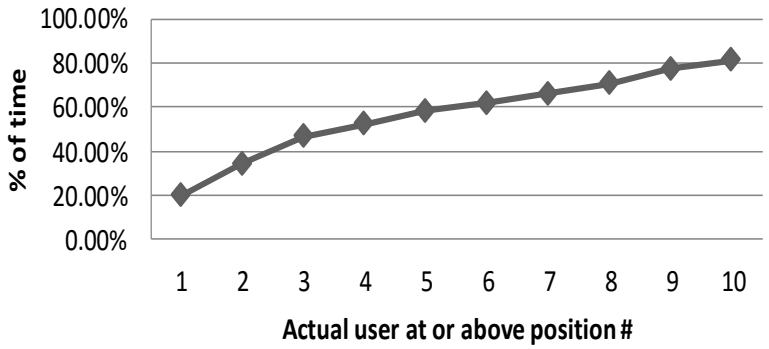

Figure 15: K-means clustering using simple ranking

In approach II, 33% of profiles, on average, were marked as possible, that is, we could eliminate 2/3 of profiles; and 83.5% of the time, the actual user is within the set of the remaining 1/3 possible profiles. Figure 16 provides an illustration of approach II performance.

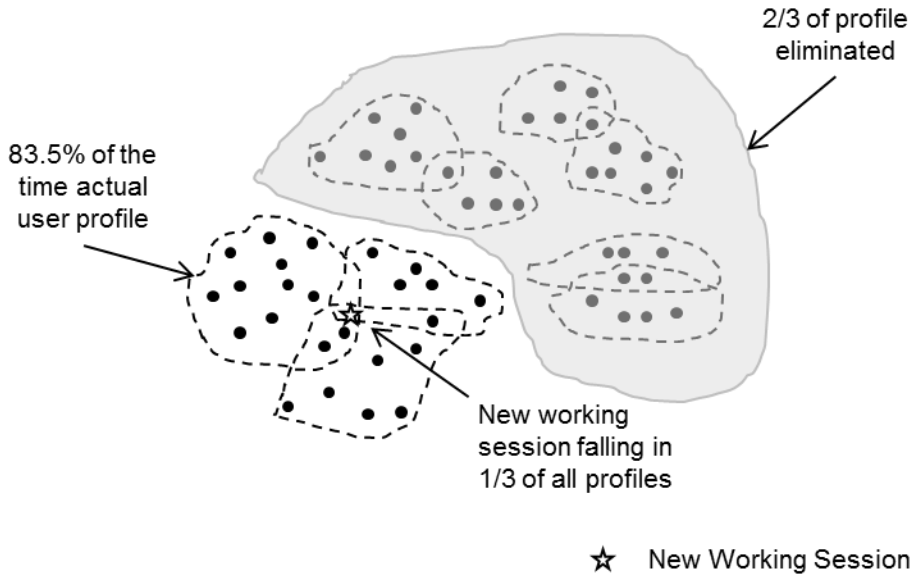

Figure 16: Illustration of k-means performance

We compare next our full-fledged authentication results to those obtained in using the screen recordings of scrolling in [43]. The best verification performance reported by [43] using SVM achieves a False Acceptance Rate (FAR) of 20.67%, a False Rejection Rate (FRR) of 12.38% and an F_1-measure Detection Error Tradeoff (DET) of 82.75%. Our all classification methods fell short in accurately predicting the authenticated user while doing really well in predicting the imposters. For example, best performance on the F_1-measure was achieved with Random Forest with sub-sequencing where we have a FAR of 0.61% and an FRR of 57.12%. The test also yielded an F_1-measure of 0.50 for the "true" class (authenticated) and 0.99 for the "false" class (imposter). This indicated that the model did well in predicting the "false" class but was neutral towards the "true" class due to the imbalanced dataset, therefore, biasing the classifier towards the majority

class (imposter). However, we do better in predicting the authenticated user with K-means clustering in which we were able to achieve a success rate of 83.5%.

Other than addressing a much more difficult problem compared to [43], authentication rather than verification, there are three additional notable differences between our experiment and the one presented in [43]. First, in [43] the number of users from which the data was collected was 21 performing 4 tasks 5 times, but only 5 users were used in testing. In our study, data was collected from 46 users (27-44 users were used with various experiments) and collectively produced 529 working sessions. 10-fold cross validation was used with classification methods, and leave-one-out cross validation with clustering. Second, the way the profiles were created in [43] was controlled with users asked to use a program that guided them through a sequence of steps that captured the four different interaction types including scrolling. In our case, the creation of users' profiles was by training the model on a subset of the dataset as it was collected from the various users' reading sessions. Third, our model is event-driven compared to time-based in [43] and therefore provide active and continuous authentication. Every scrolling event in our model triggers a point for possible authentication. In [43], the screen recording is timed without a guarantee that an actual scrolling occurred during that time. Table 15 summarizes a comparison between SBAM and other related research in terms of novelty, accuracy and significance.

Table 15: SBAM vs. Others

Model	Novelty	Accuracy		Significance	
		Research by	Results	# Users	# Sessions
SBAM	**Document Scrolling Behavior** Based on event-driven temporal data captured through monitoring users' scrolling habits User interaction uncontrolled	El-Masri et al. (2015)	84.50%	46	529
Others	**Screen fingerprint** Based on pixel-level screen analysis of captured screen recordings. User interactions controlled (program guided user through a sequence of steps)	Fathy et al. (2014)	83.50%	21	105

CHAPTER 8: APPLICATIONS AND FUTURE WORK

The provided models can lend themselves to applications in areas such as access control, intrusion detection, and recommender systems. This chapter discusses possible applications and future work.

8.1 Document Access Control

The approach for authenticating users based on their interaction with documents is suited in situations when limiting access to such documents is desired beyond the traditional access controls. While initial authentication serves as an entry point of user verification, it falls short of guaranteeing that the user who was initially authenticated is still the same user currently interacting with the documents. Organizations wishing to restrict access to classified or confidential documents would want to employ a system that verifies the subject identity while interacting with such content-sensitive documents. Therefore, a model such as the one presented here is well suited for such situations especially that the model relies solely on the scrolling behavior of the user, which by virtue of the documents being read-only would be the only means of interaction with the documents.

8.2 Electronic Document Forensics

An offshoot of our experiment provided preliminary evidence suggesting that users' knowledge or familiarity of content can be deduced from their scrolling style. To demonstrate the application of such models to the computer forensic field, consider the following pilot experiment and its results. Three users were given a three-page document containing financial data showing an organization's Income Statement, Balance Sheet and Cash Flow for the years 2008, 2009 and 2010. The first user was a novice in the subject (no background in finance and/or accounting), the second user was an expert in the subject and the third user was the author of the document. All users were given two questions and asked to find the answers in this financial document while the MS Word logger logged their reading patterns (for details on this tool see section 1.5.2 Developed Tools). Data then was plotted using MS Excel and the following two graphs are generated: Figure 17 compares the document navigation trend among the three different users by showing changes in document vertical percent scroll over time. Figure 18 provides a detailed analysis showing the vertical scroll percent over pages and the time elapsed between successive scrolls for the novice user.

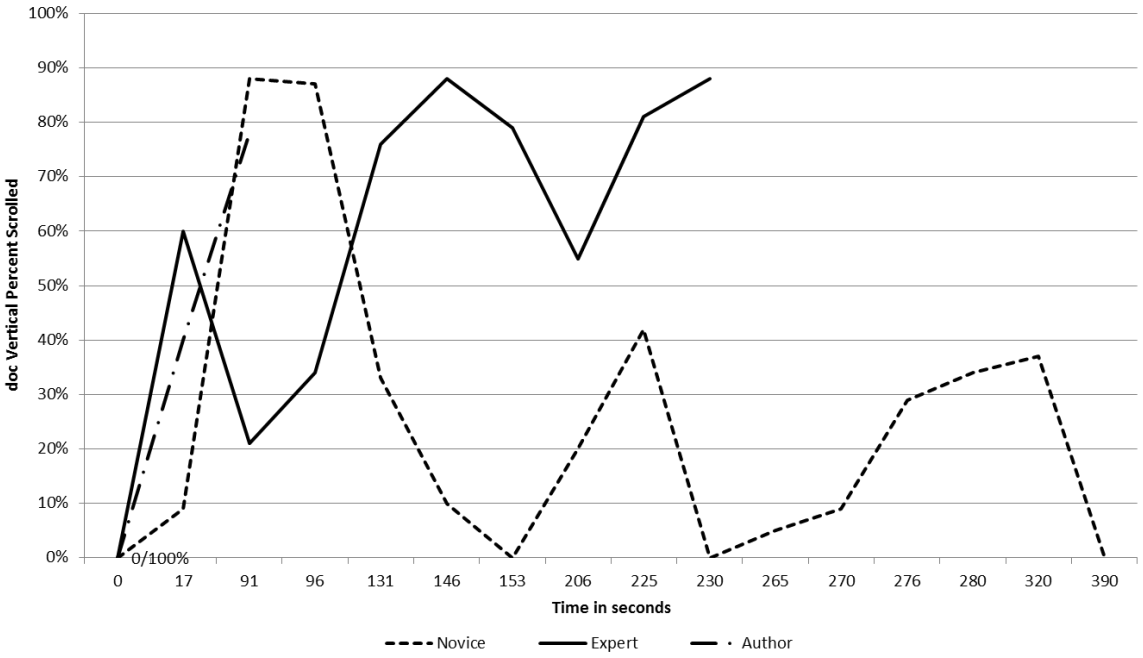

Figure 17: Document vertical scroll percent over time

To demonstrate the implications of such graphs consider the following scenario. Company xyz discovers that some classified information was leaked to the public. A security audit reveals that the source of this leaked information is a document authored by employee Bob. Access to this document is limited only to Bob, Bob's coworker and a contributor to the document's content Alice, and Bob's supervisor Mary. The audit also shows the document's access metadata such as the document access time and accessed-by properties. The metadata reveals that Bob had accessed the document numerous times; while Alice had accessed it only few times before. Mary, the supervisor, had only accessed the document only twice. Any of the three privileged users could have undermined the confidentiality of this document. Their access credentials could have

been compromised and an attacker could have carried the breach without being detected. In such scenario, it is difficult to identify the true attacker.

Bob and Alice, the coauthors, would have not browsed randomly through the document prior to the information leak. Instead, they would have directly navigated to the exact location of interest (e.g., high value content that was leaked to the public). Such navigational pattern is depicted in the expert and author series in Figure 17. On the other hand, a person who knew that the document contains critical information but does not know where exactly this information is located would have to skim randomly spending measurable time between positions in the document. This pattern is depicted in the novice series in Figure 17. Considering her limited access history and depending on her familiarity with the document, this could be Mary's, the supervisor, navigation pattern illustrated by the novice series on the graph.

Following an induction by elimination approach, Figure 18 provides a more granular analysis. It depicts Bob's, the author, page scrolling pattern and the elapsed time between scrolls revealing information such as how much time Bob spent at a particular section in a page. Assuming that the leaked information resides at location x in the breached document, if the navigation pattern does not show that the Bob has navigated to and spent time at location x, he could be eliminated as a possible attacker. While it is true that Bob could have relied on his memory or notes to leak this information, if the data shows that the user has navigated to and spent considerable time at the location of interest, again, it is less likely he is the perpetrator. Similarly, the same analysis can be applied to Alice and Mary; the other two users who had accessed the document.

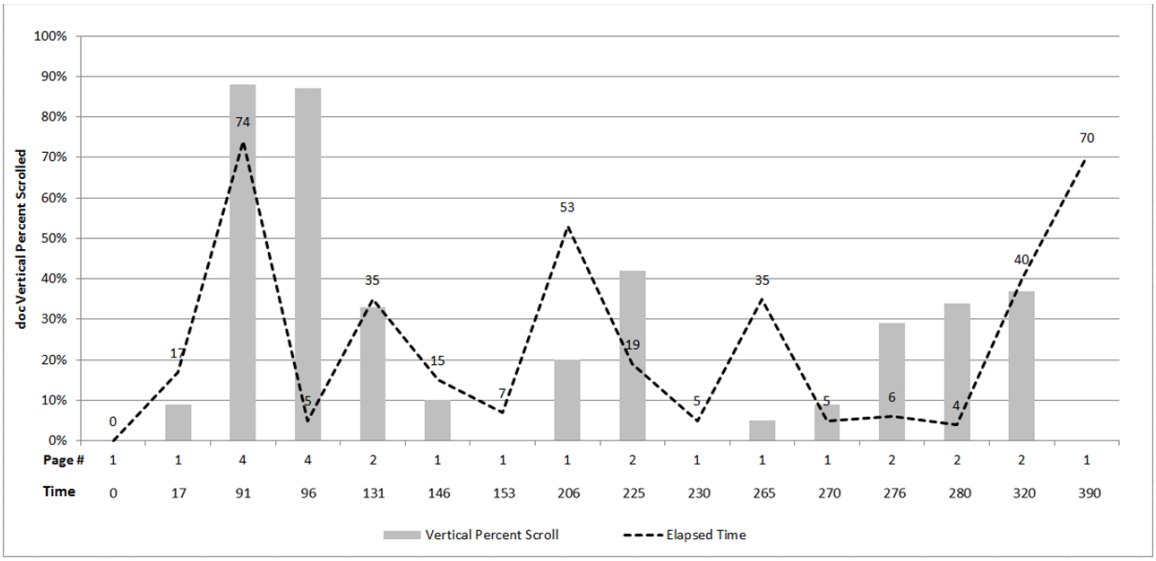

Figure 18: Document vertical scroll percent over elapsed time and page number

One can see how such model can be a useful tool in electronic document forensics. While the results presented above are empirical in nature, they serve as a preliminary evidence of the applicability of our model in the area of computer forensics and a motivation for future work.

8.3 Recommender Systems

Another offshoot of our experiments provided preliminary evidence suggesting that users' intentions can be deduced from their scrolling style. Three main styles of reading, or scrolling, were observed: skimming, scanning and careful reading. These different styles of reading can be clearly observed in Figure 19, and can be used to derive intentions behind ongoing reading allowing the system to respond accordingly. For

example, an observation suggesting a user is involved in careful reading can prompt the system to suggest further readings by providing external references to the topics being covered in that particular section.

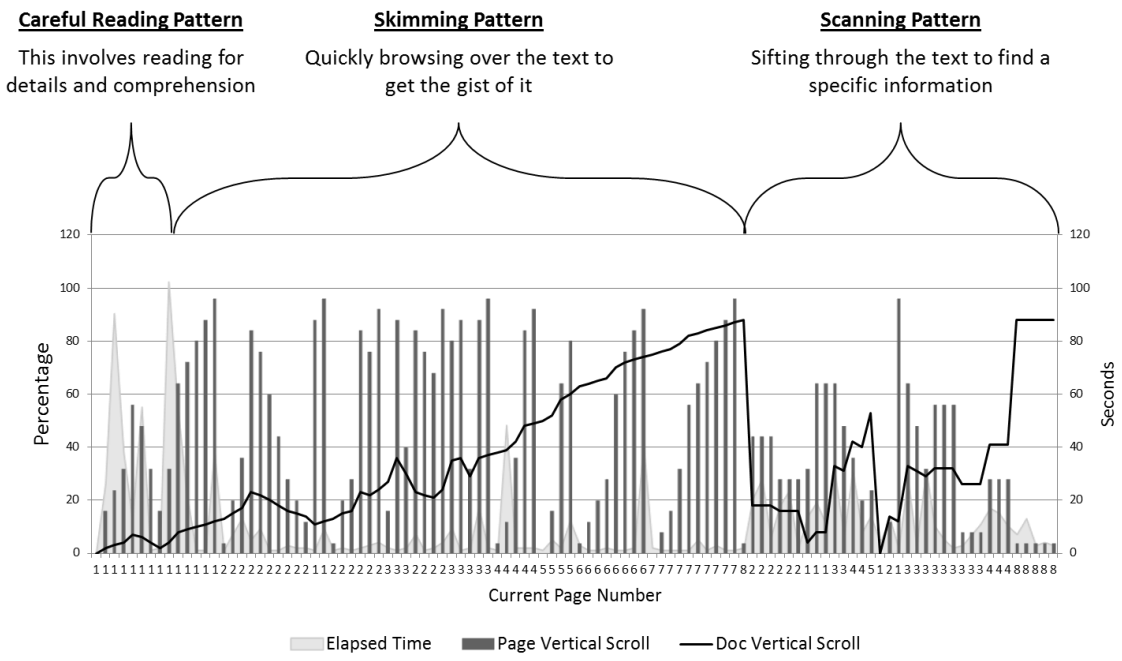

Figure 19: Observed reading patterns

The three reading styles can be determined by certain properties such as time elapsed and vertical scrolling patterns. For example, during the careful reading pattern, as illustrated in Figure 20, one can see that considerable time was spent on the first page when compared to the rest of the pages in the document. Moreover, the peaks and dips in

the page vertical scroll percentage are occurring on the same page. This is attributed to the document's two-column layout.

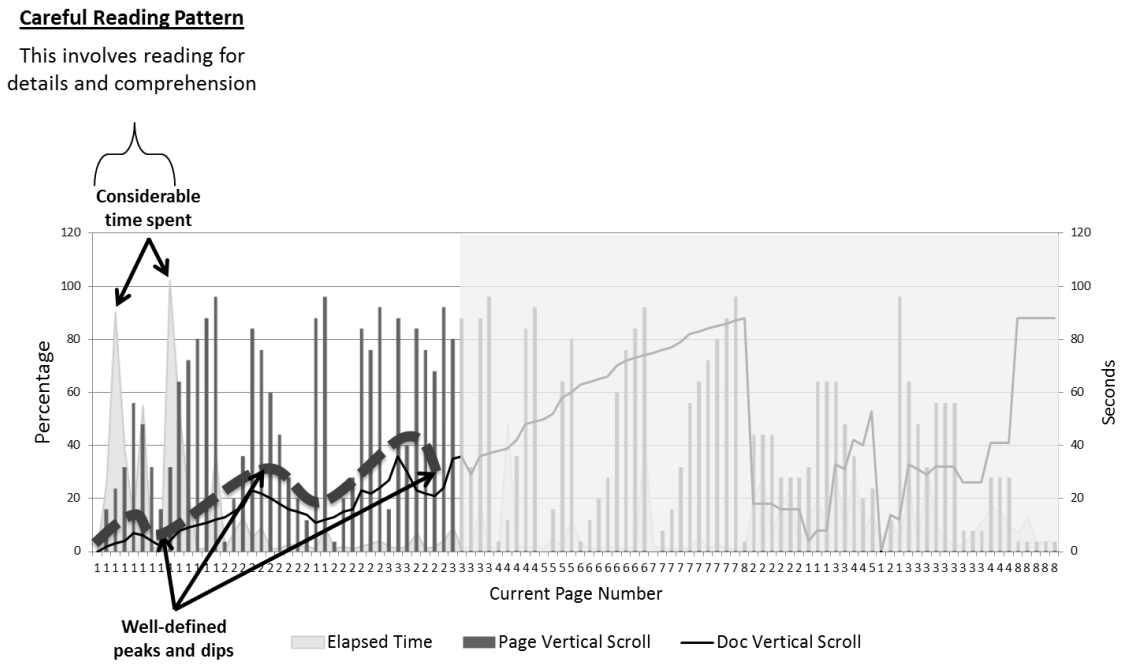

Figure 20: Careful reading pattern

However, as the user moves to a skimming reading pattern both the time elapsed between scrolls starts to decrease (an indication that the user is no longer engaged in careful reading pattern) and the peaks and dip start to gradually lose their definition (an indication that the user is no longer reading each column on the page). Another interesting observation is that during this phase the user did spend some measurable time at two particular locations in the document: page 4 at 39% vertical scroll and page 6 at

73% vertical scroll. Cross referencing the document for these locations it appears that figures and tables contributed to the longer elapsed time between scrolls.

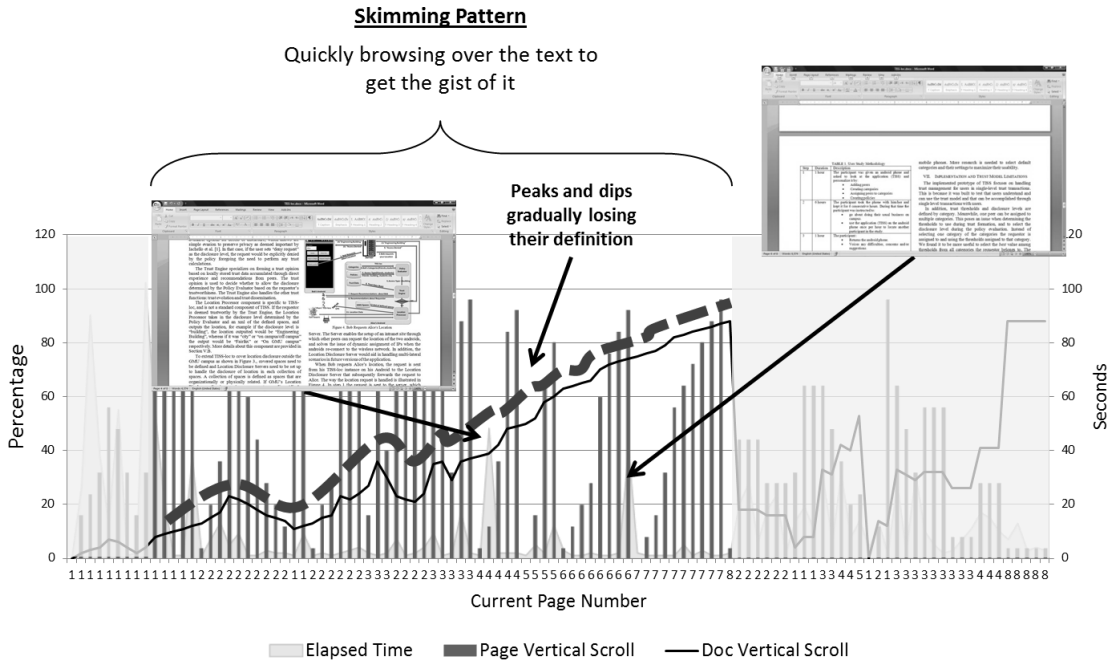

Figure 21: Skimming reading pattern

Finally, during the last phase scrolling becomes more random with the user scrolling back and forth between pages and revisiting pages. However, unlike the skimming reading pattern the user is pending more time between scrolls (12 seconds on average). This is an indication that the user is involved in a scanning reading pattern in which the goal is to find specific information.

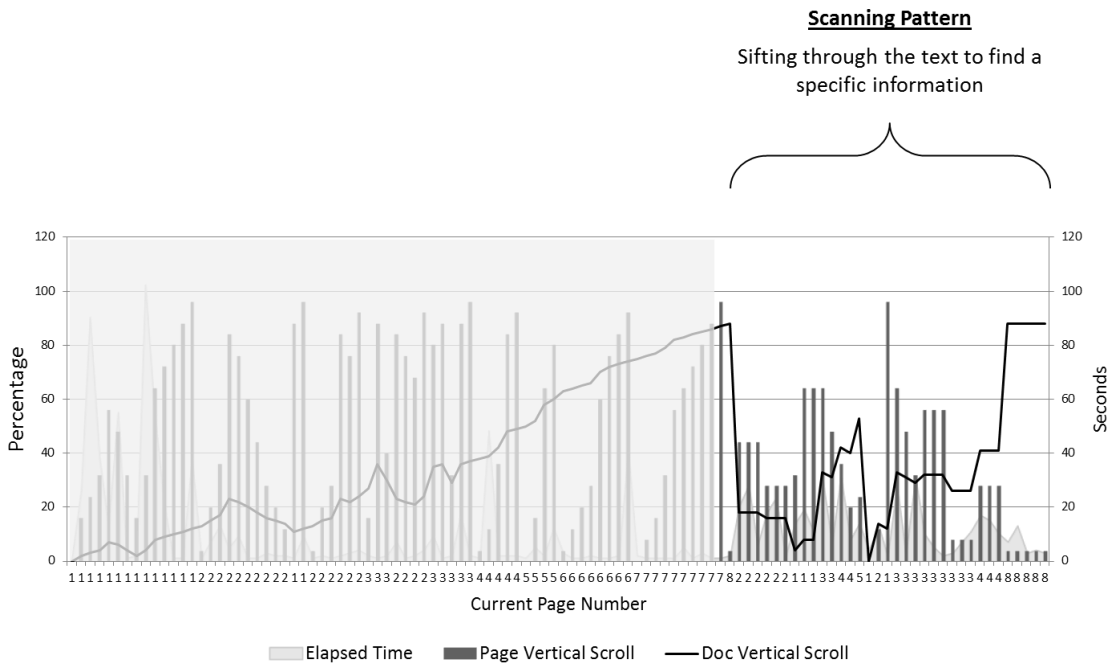

Figure 22: Scanning reading pattern

8.4 Mobile Applications

Another suitable application of our model is in mobile devices security. While many consumer market research are available for reference, it needs no proof that touchscreen devices are inarguably reaching critical mass. However, the increased popularity comes as no surprise, especially when considering what these relatively small devices can provide their users. For example, the mobile phones have evolved from simple point-to-point wireless communication devices that transmit and receive audio to fully equipped intelligent personal gadgets that are used to carry out information-sensitive activities such as online shopping and banking – which consequently raises privacy concerns and the need to protect it.

While our feature vector was composed of mainly users' scrolling traits by means of a mouse or a keyboard, it can also be applied to scrolling based on touch gestures. The wide spread of touch-oriented devices today such as smartphones and tablets are well-suited candidates. In such devices scrolling is the main interaction a user has with the GUI, and hence is a natural extension to our experiment and a fitted application to our model.

8.5 KAIST Research

Our experiment protocol and MS logger program were leveraged by a research group from KAIST to carryout similar experiments with slight modifications. Twenty-five subjects participated in this experiment. Twenty of them were graduate students who took the same class, and five participants were researchers belonging to a National Research Institute. Each individual was required to reserve a time slot to access the test machine. Three 23 page aligned column documents were used in the experiment; all written in Korean. One document was from a book chapter related to cryptography, while the other documents were from Maeil Economy, a prominent Korean economy magazine. Articles were selected based on their ease of readability and the amount of figures and tables they contain. Topics of three documents included public key encryption, funeral culture and social commerce.

To motivate participants to read the documents honestly, they were required to answer ten questions after reading each document. Each participant was asked to access the test machine, read two documents and answer two sets of related questions. Participants were divided into two groups. Ten participants in Group 1 were asked to read

document A and document B and answer the questions related to these documents. Group 1 participants had background knowledge in cryptography; the topic of document A, but no background knowledge related to document B topic. The remaining fifteen participants in Group 2 were asked to read document B and document C and answer the questions related to these documents. Participants in Group 2 did not have background knowledge related to either document.

Users' profiles were created from collected data per reading session. A reading session is defined as a set of observed events occurring on a particular page. For example, events occurring on page one are collected and treated as belonging to one reading session. Events occurring on page two are collected and treated as belonging to another reading session. If the user scrolls back to page one, the new events on this page are collected and treated as belonging to yet another reading session (not the same session defined when the user visited page one previously). Unlike how we define a reading session by building feature vector over the entire document, they define a reading session over a page. This allowed for more sessions to be defined and users' profile to be built faster (not having to wait for the entire document to be read). It also relieved users from reading multiple documents, which may lead to users' anxiety and poor participation.

To classify users as *authenticated* vs. *imposters*, four classification algorithms were used: Naïve Bayes, Support Vector Machine (SVM), classification and regression trees (CART) and Random Forest. Supervised machine learning with 10-folds cross validation was conducted in which the dataset was divided into ten distinctive folds. Each fold was used once to test the classifier while the other remaining folds were reserved for training

the classifier. This process was repeated ten times. In case of the Random Forest algorithm, the classifier was trained with 10 decision trees trained on a randomly selected subset of three features. The following general protocol was used with all classifiers:

For each user *i*:

1) Label sessions corresponding to *i* as *authenticated*
2) Label sessions not corresponding to *i* as *imposters*
3) Perform a 10-fold cross validation (CV) for each classifier algorithm
4) for j^{th} fold where j=1,…, 10:
    4.1) Train classifier with all folds except j^{th} fold and test
5) Evaluate the performance by calculating:
    5.1) the percentage of correctly classified sessions,
    5.2) Area Under the Curve (AUC) of ROC Curve and
    5.3) both false negative rate and false positive rate.

Table 16 presents the average performance of algorithms used in KAIST main experiment. Each individual's profile was treated as authenticated user once while other profiles were treated as imposters. The performance for linear classifier algorithms such as Naïve Bayes and SVM yielded high false positive rate while achieving very low false negative rate when compared to CART and Random Forest. This is due to the low dimensionality of the feature set (only three features), and hence linear classifiers could not determine proper hyperplane well.

Table 16: Comparison of average performance of algorithms used in KAIST main experiment

Algorithm	False Positive Rate	False Negative Rate	AUC
Naïve Bayes	0.74%	13.67%	0.78
SVM	0%	13.80%	0.50
CART	3.57%	4.93%	0.95
Random Forests	1.11%	2.34%	0.99

Overall, the false positive rate (the percentage of *imposters* misclassified as *authenticated* users) was relatively low compared to the false negative rate (the percentage of *authenticated* users misclassified as *imposters*). Random Forests outperformed the other algorithms (Naïve Bayes, SVM and CART) in terms of false positive and false negative rates. This is due to the randomness and bagging characteristics of Random Forests, which helps avoid overfitting.

CHAPTER 9: CONCLUSIONS

This research addresses the problem of active, also known as continuous, authentication. It does so by highlighting the need for and motivation behind such an authentication model and the limitation and challenges of existing active authentication techniques. It introduces two novel models for non-intrusive active authentication: a GUI-based application command streams model and a document scrolling behavior model.

In the GUI-based applications command stream model, we demonstrate that like a command line lexicon, GUI application command patterns can be used to create user profiles that are unique and identifiable. Past research has explored the idea of building user profiles based on users' behavioral patterns when interacting with graphical interfaces by analyzing the users' keystroke and/or mouse dynamics. However, none had explored the idea of creating profiles by capturing users' issued commands when interacting with a specific application, which goes beyond how a user strikes the keyboard or moves the mouse across the screen. It provides more dimensions to consider and a richer set of behavioral features to include when building behavioral profiles.

We repurpose a publicly available dataset (see section 6.1 Dataset) of user command streams generated from MS Word usage to serve as a test bed. User profiles are first built using MS Word commands and discrimination takes place using machine

learning algorithms. We report best performance using Random Forests and AdaBoost with Random Forests coming first in terms of both accuracy and Area under the Curve (AUC) for the Receiver Operating Characteristic (ROC) curve. This was due, first, to implementing ensemble methods, with Random Forest also characterized by a potent mix of randomness and subsampling vis-a-vis both the data samples chosen for training and the features chosen to represent the user profiles engaged in training. This is an essential adaptation strategy for active authentication to cope better with the varying nature of individual user profiles. The training strategy is further enhanced using SMOTE to handle unbalanced populations, with imposters less prevalent.

In the document scrolling behavior model, we have repurposed a dataset that was obtained from a previous research project at the Georgia Institute of Technology (see section 7.1 Dataset). Three different feature vectors were extracted and used with different classification and clustering algorithms. Our main contribution comes in the features considered, the technique used to enrich weak discriminating features and methods for continuous authentication. We relied only on the scrolling properties of the user over a document, which, in our case, was the only possible interaction a user could have with the read-only documents. We have applied Random Forests as the main classification model and used SMOTE and AdaBoost to overcome the class imbalance problem. Furthermore, we have uniquely applied n-gram sequencing technique to model the feature vector. The n-gram method was used to create the feature vector where each n-gram represents a feature corresponding to a user's scroll sequence. However, best

performance was achieved using k-means clustering with an 83.5% success rate in predicting the authenticated user.

APPENDIX A: UNIQUE COMMANDS ISSUED BY USERS IN THE MITRE DATASET

The MITRE dataset was imported into an MS Access table and the following 172 unique user issued commands where queried and retrieved:

COMMANDS			
EditAutoText	FormatBorderInside	InsertCrossReference	ToolsCustomize
EditBookmark	FormatBorderLeft	InsertDatabase	ToolsEnvelopesAndLabels
EditClear	FormatBorderLineStyle	InsertDateField	ToolsGrammar
EditCopy	FormatBorderNone	InsertDateTime	ToolsHyphenation
EditCopyAsPicture	FormatBorderOutside	InsertField	ToolsLanguage
EditCut	FormatBorderRight	InsertFile	ToolsOptions
EditDeleteBackWord	FormatBordersAndShading	InsertFootnote	ToolsProtectDocument
EditDeleteWord	FormatBorderTop	InsertFormField	ToolsRepaginate
EditFind	FormatBulletsAndNumbering	InsertFrame	ToolsRevisions
EditGoBack	FormatCenterPara	InsertObject	ToolsShrinkToFit
EditGoTo	FormatChangeCase	InsertPageBreak	ToolsSpelling
EditGoToHeaderFooter	FormatColumns	InsertPageNumbers	ToolsThesaurus
EditPaste	FormatDrawingObject	InsertPicture	ToolsWordCount
EditPasteSpecial	FormatDropCap	InsertSectionBreak	ViewAnnotations
EditRedo	FormatFont	InsertSelectDrawingObjects	ViewBorderToolbar
EditReplace	FormatFrame	InsertSymbol	ViewCloseViewHeaderFooter
EditSelectAll	FormatGrowFont	InsertTextFormField	ViewDrawingToolbar
EditUndo	FormatGrowFontOnePoint	InsertTimeField	ViewFootnotes
EditUpdateSource	FormatHangingIndent	TableAutoFormat	ViewGoToHeaderFooter
FileClose	FormatHeadingNumbering	TableColumnSelect	ViewHeader
FileClosePreview	FormatIndent	TableDeleteCellsArgs	ViewLockDocument
FileDocClose	FormatItalic	TableDeleteCellsArgs=	ViewMasterDocument
FileDocumentStatistics	FormatJustifyPara	TableDeleteColumn	ViewNormal
FileExit	FormatLeftPara	TableDeleteRow	ViewOutline
FileFind	FormatParagraph	TableFormatCell	ViewPage
FileMagnifier	FormatPicture	TableFormula	ViewRuler
FileNew	FormatResetChar	TableGridlines	ViewShowAll
FileNewDefault	FormatRightPara	TableHeadings	ViewShowAllHeadings

FileOpen
FilePageSetup
FilePrint
FilePrintDefault
FilePrintPreview
FileQuit
FileSave
FileSaveAll
FileSaveAs
FileSummaryInfo
FileTemplates
FormatAllCaps
FormatAutoFormat
FormatBold
FormatBorderBottom
FormatShrinkFontOnePoint
FormatStrikethrough
FormatStyle
FormatStyleGallery
FormatSubscript
FormatTabs
FormatUnderline
FormatUnindent
FormatWordUnderline
Help
HelpAbout
HelpTipOfTheDay
HelpTool
InsertAnnotation
InsertBreak
TableInsertColumn
TableInsertRow
TableInsertTable
TableMergeCells
TableSelectColumn
TableSelectRow
TableSelectTable
TableSort
TableSortAToZ
TableSortZToA
TableSplit
TableSplitCells
TableToOrFromText
ToolsAddRecordDefault
ToolsCancle
ViewShowNextHeaderFooter
ViewShowPrevHeaderFooter
ViewToggleFull
ViewToggleMasterDocument
ViewToolbars
ViewZoom
ViewZoomWholePage
WindowArrangeAll
WindowClosePane
WindowDocMaximize
WindowDocMinimize
WindowDocRestore
WindowDocSplit
WindowNewWindow
WindowShowClipboard

APPENDIX B: SAMPLE DATA FROM THE MITRE DATASET

The MITRE Corporation makes available through its research web portal at http://www.cs.rutgers.edu/ml4um/ a dataset of approximately 74,000 records that were acquired by monitoring day to day usage of Microsoft Word by more than 20 individuals at the MITRE Corporation during the calendar years of 1997-1998. The users consisted of artificial intelligence engineers, technical staff, and support staff. The following table describes the data collected:

Data Description for MITRE dataset

Column Label	Description
user ID	Unique identifier for each user
version of Word	Version of Word in use at the time of logging
file size	The size of the file, in bytes, when logging was initiated
file date	The date logging was initiated for that file
operating system and version	The operating system and version in use at the time of logging
command name	The name of the command the user entered. The command name is proceeded by the type of command, i.e., *EditSelectAll* is the text editing command *SelectAll* nominally appearing on the Edit menu.
command date	The date the user entered the command
command time	The time the user entered the command

File size allows researchers to correlate commands used with file size. File size and date may be joined to form a unique file identifier, allowing researchers to correlate commands used with individual files. A command may be entered by selecting it from a menu, clicking on an icon, or pressing a keyboard combination. The logger does not distinguish how commands are entered. The following is a sample data from the MITRE dataset (the complete set can be obtained at http://www.cs.rutgers.edu/ml4um/):

Sample data from the MITRE dataset

USER-ID	WORD-VERSION	FILE-DATE	FILE-SIZE	MACHINE-OS	COMMAND	COMMAND-DATE	COMMAND-TIME
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	11:54:12 AM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	11:54:12 AM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	FileSave	6/2/1998	11:54:17 AM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	FilePrintDefault	6/2/1998	11:54:21 AM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	FileOpen	6/2/1998	5:05:03 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	FileOpen	6/2/1998	5:05:05 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:05:42 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	FileSaveAs	6/2/1998	5:10:34 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:16:01 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:19:27 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:19:27 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:19:27 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:19:27 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:19:28 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:19:44 PM
21464	Version	6/2/1998	903	Macintosh	EditClear	6/2/1998	5:19:44 PM

	6.0.1			7.5.3			
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:19:44 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:19:45 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:19:45 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:19:45 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:19:46 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:24:52 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	EditClear	6/2/1998	5:25:02 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	FileSave	6/2/1998	5:25:45 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	FileSave	6/2/1998	5:27:09 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	FilePrintDefault	6/2/1998	5:27:11 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	FilePrintDefault	6/2/1998	8:49:35 PM
21464	Version 6.0.1	6/2/1998	903	Macintosh 7.5.3	FileDocClose	6/2/1998	9:35:58 PM
21464	Version 6.0.1	6/2/1997	"4,691"	Macintosh 7.5.3	EditCut	6/12/1998	9:48:28 AM
21464	Version 6.0.1	6/2/1997	"4,691"	Macintosh 7.5.3	EditPaste	6/12/1998	9:48:34 AM
21464	Version 6.0.1	6/2/1997	"4,691"	Macintosh 7.5.3	FileSaveAs	6/12/1998	9:55:19 AM
21464	Version 6.0.1	6/2/1997	"4,691"	Macintosh 7.5.3	FilePrint	6/12/1998	10:04:59 AM
21464	Version 6.0.1	6/2/1997	"4,691"	Macintosh 7.5.3	FilePrint	6/12/1998	10:06:11 AM
21464	Version 6.0.1	6/2/1997	"4,691"	Macintosh 7.5.3	FileSave	6/12/1998	10:10:43 AM
21464	Version 6.0.1	6/2/1997	"4,691"	Macintosh 7.5.3	FilePrint	6/12/1998	10:10:51 AM
21464	Version 6.0.1	6/2/1997	"4,691"	Macintosh 7.5.3	FileSave	6/12/1998	11:15:18 AM
21464	Version 6.0.1	6/2/1997	"4,691"	Macintosh 7.5.3	FilePrint	6/12/1998	11:18:09 AM
21464	Version 6.0.1	6/2/1997	"4,691"	Macintosh 7.5.3	FileSave	6/12/1998	2:36:07 PM
21464	Version 6.0.1	6/2/1997	"4,691"	Macintosh 7.5.3	FilePrint	6/12/1998	2:36:14 PM
21464	Version 6.0.1	5/29/1997	0	Macintosh 7.5.3	FileOpen	6/15/1998	2:34:59 PM
21464	Version 6.0.1	5/29/1997	0	Macintosh 7.5.3	FileDocClose	6/15/1998	2:35:00 PM
21464	Version	6/2/1997	"4,691"	Macintosh	FileQuit	6/16/1998	3:53:05 PM

	6.0.1			7.5.3			
21464	Version 6.0.1	6/2/1997	"4,691"	Macintosh 7.5.3	FileFind	6/16/1998	8:45:37 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	FileOpen	5/22/1997	2:48:48 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	FormatBold	5/22/1997	2:48:49 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	FormatBold	5/22/1997	2:48:53 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	FormatUnderline	5/22/1997	2:48:56 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	FormatUnderline	5/22/1997	2:48:59 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	ViewToolbars	5/22/1997	2:49:09 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	FormatCenterPara	5/22/1997	2:49:11 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	FileQuit	5/22/1997	2:50:52 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	FileOpen	5/22/1997	3:34:26 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	FormatBold	5/22/1997	3:34:27 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	FileQuit	5/22/1997	3:34:42 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	FileOpen	5/22/1997	3:37:23 PM
M000	Version 6.0.1	5/22/1997	6	Macintosh 7.5.3	FormatBold	5/22/1997	3:37:25 PM

REFERENCES

[1] Aberdeen Group, “Strong User Authentication - Best-in-Class Performance at Assuring Identites,” Mar. 2008.

[2] “2013 DATA BREACH INVESTIGATIONS REPORT,” Verizon.

[3] M. Bishop, “Position: Insider is relative,” in *Proceedings of the 2005 workshop on New security paradigms*, 2005, pp. 77–78.

[4] bitglass, “The 2014 Bitglass Heathcare Breach Report,” Bitglass, Inc., 2014.

[5] T. G. Dietterich, “Ensemble methods in machine learning,” in *Multiple classifier systems*, Springer, 2000, pp. 1–15.

[6] A. El Masri, H. Wechsler, P. Likarish, and B. B. Kang, “Identifying users with application-specific command streams,” in *Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on*, 2014, pp. 232–238.

[7] A. El Masri, H. Wechsler, P. Likarish, C. Grayson, C. Pu, D. Al-Arayed, and B. B. Kang, “Active authentication using scrolling behaviors,” in *Information and Communication Systems (ICICS), 2015 6th International Conference on*, 2015, pp. 257–262.

[8] J. Wu and Z. Chen, “An Implicit Identity Authentication System Considering Changes of Gesture Based on Keystroke Behaviors,” *Int. J. Distrib. Sens. Netw.*, vol. 2015, 2015.

[9] H. Wechsler, “Cyberspace Security using Adversarial Learning and Conformal Prediction,” *Intell. Inf. Manag.*, vol. 7, no. 04, p. 195, 2015.

[10] G. Stoneburner, *Underlying technical models for information technology security: recommendation of the National Institute of Standards and Technology*. US Department of Commerce, Computer Security Division, Information Technology, National Institute of Standards and Technology, 2001.

[11] D. M. Cappelli, A. P. Moore, and R. F. Trzeciak, *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud)*. Addison-Wesley, 2012.

[12] J. T. FORCE and T. INITIATIVE, “Security and Privacy Controls for Federal Information Systems and Organizations,” *NIST Spec. Publ.*, vol. 800, p. 53, 2013.

[13] A. Jain, L. Hong, and S. Pankanti, “Biometric identification,” *Commun. ACM*, vol. 43, no. 2, pp. 90–98, 2000.

[14] “Active Authentication.” [Online]. Available: http://www.darpa.mil/Our_Work/I2O/Programs/Active_Authentication.aspx. [Accessed: 18-May-2012].

[15] V. Piuri and F. Scotti, “Fingerprint biometrics via low-cost sensors and webcams,” in *Biometrics: Theory, Applications and Systems, 2008. BTAS 2008. 2nd IEEE International Conference on*, 2008, pp. 1–6.
[16] N. Brown, “Mobile Verification by Palmprint Biometrics.”
[17] Y. Han, T. Tan, Z. Sun, and Y. Hao, “Embedded palmprint recognition system on mobile devices,” *Adv. Biom.*, pp. 1184–1193, 2007.
[18] K. W. Bowyer, K. Hollingsworth, and P. J. Flynn, “Image understanding for iris biometrics: A survey,” *Comput. Vis. Image Underst.*, vol. 110, no. 2, pp. 281–307, 2008.
[19] T. Ahonen, A. Hadid, and M. Pietikainen, “Face Description with Local Binary Patterns: Application to Face Recognition,” *IEEE Trans. PATTERN Anal. Mach. INTELL*, vol. 28, pp. 2037 – 2041, Dec. 2006.
[20] F. Li and H. Wechsler, “Open set face recognition using transduction,” *Pattern Anal. Mach. Intell. IEEE Trans. On*, vol. 27, no. 11, pp. 1686–1697, 2005.
[21] Y. Rodriguez and S. Marcel, “Face Authentication Using Adapted Local Binary Pattern Histograms,” presented at the 9th European Conference on Computer Vision ECCV, 2006.
[22] A. J. Klosterman, “Secure continuous biometric-enhanced authentication,” DTIC Document, 2000.
[23] D. R. KISKU, P. GUPTA, H. MEHROTRA, and J. K. SING, “Multimodal Belief Fusion for Face and Ear Biometrics,” *Intell. Inf. Manag.*, vol. 1, no. 3, pp. 166–171, 2009.
[24] S. Noushath, M. Imran, K. Jetly, A. Rao, and G. Hemantha Kumar, “Multimodal biometric fusion of face and palmprint at various levels,” in *Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on*, 2013, pp. 1793–1798.
[25] A. C. Morris, S. Jassim, H. Sellahewa, L. Allano, J. Ehlers, D. Wu, J. Koreman, S. Garcia-Salicetti, B. Ly-Van, and B. Dorizzi, “Multimodal person authentication on a smartphone under realistic conditions,” in *Proc. SPIE conference on mobile multimedia/image processing for military and security applications, Orlando*, 2006.
[26] A. Azzini, S. Marrara, R. Sassi, and F. Scotti, “A fuzzy approach to multimodal biometric continuous authentication,” *Fuzzy Optim. Decis. Mak.*, vol. 7, no. 3, pp. 243–256, 2008.
[27] R. H. . Yap, T. Sim, G. X. . Kwang, and R. Ramnath, “Physical Access Protection using Continuous Authentication,” in *2008 IEEE Conference on Technologies for Homeland Security*, 2008, pp. 510–512.
[28] M. Karnan, M. Akila, and N. Krishnaraj, “Biometric personal authentication using keystroke dynamics: A review,” *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1565–1573, 2011.
[29] R. Katiyar, V. K. Pathak, and K. V. Arya, “A Study on Existing Gait Biometrics Approaches and Challenges.,” *Int. J. Comput. Sci. Issues IJCSI*, vol. 10, no. 1, 2013.
[30] I. Traore, I. Woungang, Y. Nakkabi, M. S. Obaidat, A. A. E. Ahmed, and B. Khalilian, “Dynamic Sample Size Detection in Learning Command Line Sequence

for Continuous Authentication," *Syst. Man Cybern. Part B Cybern. IEEE Trans. On*, vol. PP, no. 99, pp. 1 –14, 2012.
[31] D. Shanmugapriya and G. Padmavathi, "A survey of biometric keystroke dynamics: Approaches, security and challenges," *Arxiv Prepr. ArXiv09100817*, 2009.
[32] A. A. E. Ahmed and I. Traore, "A new biometric technology based on mouse dynamics," *Dependable Secure Comput. IEEE Trans. On*, vol. 4, no. 3, pp. 165–179, 2007.
[33] A. Weiss, A. Ramapanicker, P. Shah, S. Noble, and L. Immohr, "Mouse movements biometric identification: A feasibility study," *Proc Stud. Res. Day CSIS Pace Univ. White Plains NY*, 2007.
[34] S. Hashiaa, C. Pollettb, M. Stampc, and M. Q. Hall, "On using mouse movements as a biometric," 2005.
[35] K. Revett, H. Jahankhani, S. T. Magalhaes, and H. M. D. Santos, "A survey of user authentication based on mouse dynamics," *Glob. E-Secur.*, pp. 210–219, 2008.
[36] A. A. E. Ahmed and I. Traore, "Detecting computer intrusions using behavioral biometrics," in *Third Annual Conference on Privacy, Security and Trust, St. Andrews, New Brunswick, Canada*, 2005.
[37] A. Garg, S. Vidyaraman, S. Upadhyaya, and K. Kwiat, "USim: a user behavior simulation framework for training and testing IDSes in GUI based systems," in *Simulation Symposium, 2006. 39th Annual*, 2006, p. 8–pp.
[38] T. Lane and C. E. Brodley, "Sequence matching and learning in anomaly detection for computer security," in *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, 1997, pp. 43–49.
[39] J. Marin, D. Ragsdale, and J. Sirdu, "A hybrid approach to the profile creation and intrusion detection," in *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, 2001, vol. 1, pp. 69–76.
[40] D. Y. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern Recognit.*, vol. 36, no. 1, pp. 229–243, 2003.
[41] M. Schonlau, W. DuMouchel, W. H. Ju, A. F. Karr, M. Theus, and Y. Vardi, "Computer intrusion: Detecting masquerades," *Stat. Sci.*, pp. 58–74, 2001.
[42] R. A. Maxion and T. N. Townsend, "Masquerade detection using truncated command lines," in *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, 2002, pp. 219–228.
[43] M. E. Fathy, V. M. Patel, T. Yeh, Y. Zhang, R. Chellappa, and L. S. Davis, "Screen-based active user authentication," *Elsevier*, vol. 42, no. Pattern Recognition Letters, pp. 122 – 127, 2014.
[44] P. N. Tan, M. Steinbach, V. Kumar, and others, *Introduction to Data Mining*. Pearson Addison Wesley Boston, 2006.
[45] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.
[46] F. Linton, D. Joy, H. P. Schaefer, and A. Charron, "Owl: A recommender system for organization-wide learning," *Educ. Technol. Soc.*, vol. 3, no. 1, pp. 62–76, 2000.

[47] “Microsoft Office Is Right at Home.” [Online]. Available: http://www.microsoft.com/en-us/news/features/2009/jan09/01-08cesofficeqaschultz.aspx. [Accessed: 24-May-2012].
[48] P. Karcher, P. Burris, and T. Keitt, “Market Update: Office 2013 And Productivity Suite Alternatives,” Forrester Research, Inc, Oct. 2013.
[49] D. Opitz and R. Maclin, “Popular ensemble methods: An empirical study,” *J. Artif. Intell. Res.*, pp. 169–198, 1999.
[50] R. Polikar, “Ensemble based systems in decision making,” *Circuits Syst. Mag. IEEE*, vol. 6, no. 3, pp. 21–45, 2006.
[51] L. Rokach, “Ensemble-based classifiers,” *Artif. Intell. Rev.*, vol. 33, no. 1–2, pp. 1–39, 2010.
[52] J. A. Quinn, C. K. Williams, J. Martí, J. M. Benedí, A. M. Mendonça, and J. Serrat, “Pattern Recognition and Image Analysis: Third Iberian Conference, IbPRIA 2007, Girona, Spain, June 6-8, 2007, Proceedings, Part I,” 2007.
[53] C. Pu and C. Grayson, “Fine-Grain Document Access Modeling and Monitoring,” Georgia Technoloby Research Corporation, Atlanta, 2012.

BIOGRAPHY

Ala'a El Masri received his Bachelor of Science in Computer Science from Costal Carolina University in 2002 and a minor in Business Administration. In 2006, he received his Master of Science in Information Technology from the University of North Carolina at Charlotte.