Winter 2020

Poisoning Attacks on Learning-Based Keystroke Authentication and a Residue Feature Based Defense

Zibo Wang

POISONING ATTACKS ON LEARNING-BASED KEYSTROKE AUTHENTICATION AND A RESIDUE FEATURE BASED DEFENSE

by

Zibo Wang M.S.

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

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

March 2020

Replace this page with the Signature Page.

ABSTRACT

Behavioral biometrics, such as keystroke dynamics, are characterized by relatively large variation in the input samples as compared to physiological biometrics such as fingerprints and iris. Recent advances in machine learning have resulted in behavior-based pattern learning methods that obviate the effects of variation by mapping the variable behavior patterns to a unique identity with high accuracy. However, it has also exposed the learning systems to attacks that use updating mechanisms in learning by injecting imposter samples to deliberately drift the data to impostors' patterns. Using the principles of adversarial drift, we develop a class of poisoning attacks, named Frog-Boiling attacks. The update samples are crafted with slow changes and random perturbations so that they can bypass the classifiers detection. Taking the case of keystroke dynamics which includes motoric and neurological learning, we demonstrate the success of our attack mechanism. We also present a detection mechanism for the frog-boiling attack that uses correlation between successive training samples to detect spurious input patterns. To measure the effect of adversarial drift in frog-boiling attack and the effectiveness of the proposed defense mechanism, we use traditional error rates such as FAR, FRR, and EER and the metric in terms of shifts in biometric menagerie.

Replace this page with the approval for scholarly dissemination form.

DEDICATION

I dedicate this dissertation to my wife and daughters. Thanks to their support along the journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation and Overview

Motivated by the abundance of sensors built into computing devices these days, there is now a wide range of research studies fronting behavioral biometrics modalities that leverage data generated by these sensors. This family of behavioral biometric modalities includes modalities such as gait [4] and touch dynamics [16] on smartphones and desktop-centric modalities such as keystroke dynamics [26] and mouse dynamics [3]. Different from physical biometric modalities such as fingerprints [22], face [10], and iris recognition [13], behavioral biometric patterns tend to be unstable [39] [28] as users generally exhibit significant variance in their patterns. While many of these biometric modalities have been shown to produce low error rates during authentication experiments conducted in controlled settings, it is not well understood how or whether the known instability of the associated behavioral biometrics patterns would impact the performance of a real authentication system in the wild.

For example, several studies have showcased error rates of less than 5% for behavioral biometric modalities such as touch dynamics [16], keystroke dynamics [6] and gait [20], to mention but a few. However, the majority of these studies were based on experiments in which data was collected from a group of users who interacted with

the system over a period of a few days. From these studies, it is hence not possible to determine how variations in users behavioral biometric patterns over time (say over several months) would impact the overall system performance. Recent efforts to thwart the potential negative impacts of these variations have showcased template update mechanisms that correct these template variations to prevent them from negatively impacting the classification engine (e.g., see [18]). However, these studies still suffer from the same problem of studying behavioral biometrics using small datasets collected over a short period, and mostly focus on showcasing the benefits of the template update schemes without providing any systematic insights into the template variations that the update mechanisms aim to fix.

Moreover, a noteworthy issue facing these template update schemes is that they have the potential to be used as a vehicle of attacks on the system i.e., along with the template update data, an adversary could insert bad data that is tuned to systematically add noise to a user's template. This problem has to some extent been studied in more established biometric modalities such as face and fingerprints [37][32], however, in behavioral biometrics, it is not known how or whether a template update scheme could be leveraged by an adversary to compromise the system. In comparison to physical biometrics such as fingerprints, behavioral biometrics exhibit much more variance, which implies that past findings on template update-related challenges seen with physical biometric modalities would not be directly applicable to a behavioral biometric setting. To protect a biometric authentication system from "adversarial drifting" type of attacks, several recent studies proposed defensive mechanisms [18, 19] to reject fake updates. However, these mechanisms also have limitations, in terms that

the updating performance is compromised, since many of the samples from genuine users are rejected from updating.

This dissertation takes steps to tackle several of the problems discussed above. In particular, we take the case of keystroke dynamics and examine the problem of feature evolution in behavioral biometric authentication. We explore the impact of feature evolution on authentication error rates and conduct a systematic evaluation of the mechanism of feature evolution over time. Armed with our findings on the nature of users feature variations, we evaluate the performance of various template update schemes and address the question of how an adversary could take advantage of the template update scheme to systematically drift a user's template. Specifically we tackle this last question through the design of a new attack called a Frog-boiling attack [38]. This attack is named after the tale of the boiling frog i.e., a frog which was unaware that it was being boiled because the water temperature had been increased in very tiny steps (see [17]). Our attack takes the same approach as the adversary aims to drift the template using very small updates that are too small to be detected, yet cumulatively able to cause a significant impact on the system over time. To prevent the attack, we propose a residual-based defense mechanism to detect and reject fake updates to a biometric system, without sacrificing and losing too many updates from the genuine users To support our investigations, we use two large datasets, one of which collected over a period of three years at Louisiana Tech University (see dataset in [35]), and the other collected over two days at Carnegie Mellon University (see [26]). The latter dataset contains data from 138 users while the former contains data from 51 users. Both datasets were collected while users typed a fixed text. Usage of

two datasets collected in different settings enables us to get deeper insights into the dynamics of template evolution.

1.2 Behavioral Biometrics

Biometrics is the technical term refers to measurements or metrics related to human characteristics. Technologies which use these measurements to verify human individuals are generally categorized under the term biometric authentication. Biometric authentication can be based on physical attributes of humans (e.g., fingerprints, face, iris patterns) or on the behavioral patterns of a human (e.g., walking patterns, typing patterns, mouse movement patterns). In both cases, a biometric system works by comparing a user's data samples with a template stored in advance. Depending on the score obtained during the comparison, a user may be rejected or accepted by the system. Where a genuine user is rejected by the system, such an event is referred to as a *false rejection* (or *false negative*) and would typically be caused by a user seeing variations in the biometric pattern relative to the stored template. Where an impostor gets accepted by the system, such an event is referred to as a *false acceptance* (or *false positive*), and occurs when the impostor has a biometric pattern similar to that of the genuine user. In general, behavioral biometrics see much more incidences of false acceptances and false rejections relative to physical biometrics. Despite their much higher susceptibility to recognition errors, behavioral biometrics retain certain advantages over physical biometrics that continue to attract a significant amount of research attention to them.

For example, physical biometrics require the user to perform a particular task (e.g., hold the finger against a fingerprint reader, face the camera for a photograph to be take, etc.). This requirement is acceptable where users have to authenticate once before accessing a resource (e.g., when logging into a network). However, where users have to be authenticated continuously, the requirement to have the user switch from the task at hand to perform the dedicated authentication task is quite intrusive. As an illustration of a situation where continuous authentication might be needed, take the case of an individual accessing confidential data on a computer (e.g., installations such as those in the Department of Defense). In order to determine that the individual accessing the computer is the legitimate user, a password is typically entered at the start of the session. However, shortly after entering the password, there is no way in which the system can determine that the user accessing the resource is still the one who provided the valid password. A naive solution to this problem would, for instance, require the user to enter the password every few minutes. For many applications, however, such an approach is bound to be rejected by users since it would perpetually distract the user away from the main task on the computer.

Behavioral biometrics offer a natural solution to such a problem since they are based on activities that the user is supposed to undertake anyway. For example, a keystroke authentication mechanism only requires the user to go about his or her routine tasks while the authentication mechanism extracts features from the typing samples. In other words, the user can be continuously authenticated without even having to be aware of, or pay attention to, the underlying authentication mechanism.

Other behavioral biometric modalities such as gait, mouse movements, and swiping patterns follow the same philosophy.

Another scenario where behavioral biometrics have interesting characteristics is that of password hardening ([31]). This scenario is most commonly studied in the breath of typing-based behavioral biometrics and basically entails the combination of a password and the typing pattern into a hardened password ([31]) that is verified at login-time. When the user types the password, the system only authenticates the user when the password, and the method of typing the password match those of the genuine user. This idea of password hardening can be extrapolated to other emerging behavioral authentication modalities such as brain activity-based authentication ([12]). Overall, these cited scenarios (i.e., continuous authentication and password hardening) illustrate why behavioral biometrics continue to be very widely studied despite being more susceptible to errors than the more established physical biometric modalities.

The challenges of behavioral biometrics authentication discussed in Section 1.1 apply to all the behavioral biometric modalities mentioned here and more. This dissertation is, however, focused only on keystroke dynamics so as to enable a thorough investigation while at the same time minimizing duplications (as would be the case if similar analysis is repeated for multiple modalities). In the following section we present an overview of keystroke dynamics.

1.3 Keystroke Dynamics

Keystroke dynamics (KD) a biometric modality in which keyboard typing patterns are used to authenticate users is categorized into fixed text verification, and

continuous text verification. In the latter category, a user's keystrokes are monitored continuously, or periodically during an entire typing session (typically after the login phase), while the former category, which is also the focus of this dissertation, is based on a fixed pre-determined text such as a password entered at login time. Verification algorithms designed for both KD branches generally draw from the same pool of features, of which the most commonly used are the key hold time (KHT), key interval time (KIT) and the keypress time (KPT) [25][35]. KHT is the time between press and release of the same key, KIT is the time between release of a key and press of the next key, and KPT is the time between press of a key and press of the next key, which equivalents to the sum of KHT of the first key and KIT of the two keys. Notice that KIT could be zero or a negative value since users may press a key on or before the previous key is released.

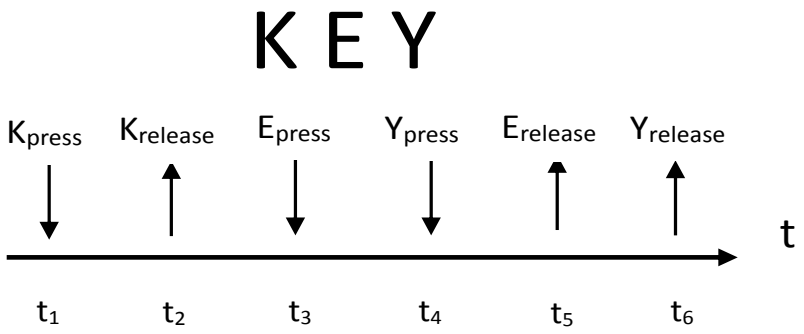


Figure 1.1: Illustrating how keystroke features are computed from an arbitrary types string. The example used here is the string KEY.

Figure 1.1 gives an example of how keystroke features are extracted when the word KEY is typed on the keyboard. A key-logger captures the time of occurrence of keypress and key release events, and keystroke features are calculated accordingly. For example, with the character K pressed at time t_1 and released at time t_2, the KHT of

key K is calculated as $t_2 - t_1$. If the next key E is pressed at t_3, the KIT and KPT of the digraph KE are $t_3 - t_2$ and $t_3 - t_1$, respectively. Notice that since key E was released after the keypress of the next key Y, the KIT of digraph EY is a negative (i.e. $t_4 - t_5$).

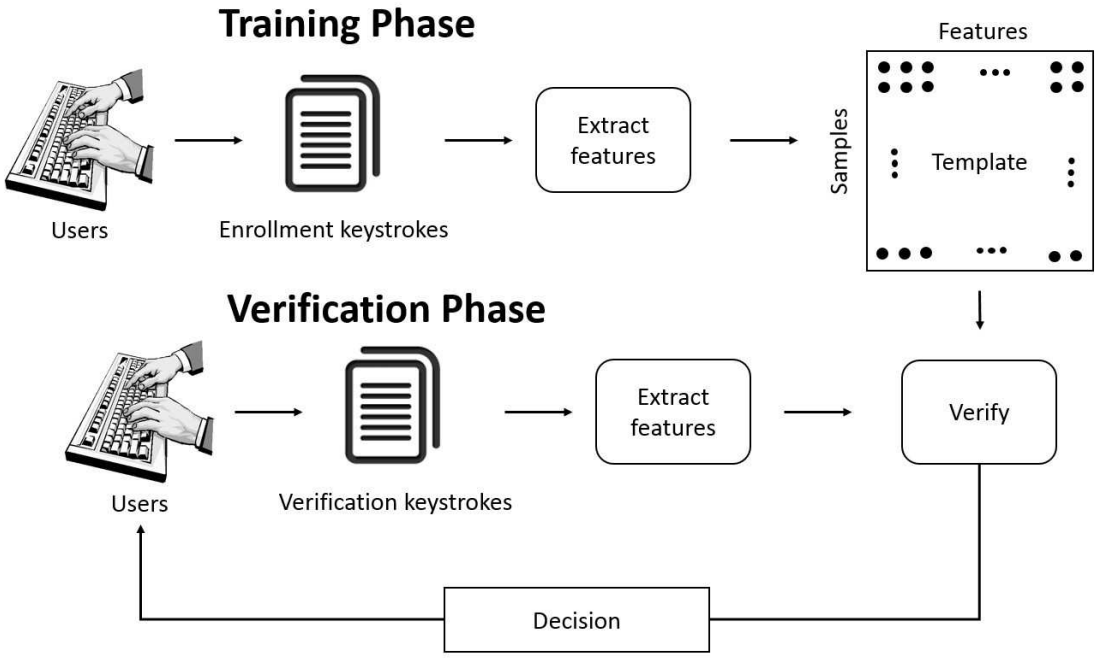


Figure 1.2: Overview of keystroke verification system on fixed text. In the training phase, keystroke features are extracted and a template (which is a matrix of order ab) is created for each user from a features collected during b typing attempts. In the verification phase, features extracted from a keystroke sample is compared with the template to produce a verification decision.

Figure 1.2 illustrates the process of a typical keystroke verification system with fixed text keystrokes. In the training phase, each user is asked to type keystrokes for enrollment, and a template is created. In the verification phase, the system compares a user's keystroke typing against the template from this user, calculates a verification score, and gives a verification decision based on a set threshold. In this dissertation,

we formalize a keystroke template as an ab matrix created from features collected during b typing attempts, and each row represents a feature vector (i.e. KHTs, KITs, and KPTs) extracted from a typing attempt.

1.4 Dissertation Contributions

The contributions of this dissertation are summarized below:

1. We analyze the variability of users keystroke biometric patterns (i.e., template evolution or aging) based on data collected over a 2-year period. In particular, we use a wide range of statistical and pattern analysis techniques to study the extent of variations and whether they are statistically significant. We perform our analysis at both feature and sample level and investigate how these variations are distributed across the population, how they impact recognition error rates and how they interact with different template update schemes. This analysis provides new insights into the design of template update schemes as well as the possibility of attacks that exploit the variations via template update schemes. To our knowledge, this is the first work to study keystroke biometric aging effect. Results of the analysis will impact research investigating template update strategies as well as motivate defenses against exploits such as the adversarial template drifts.

2. We design and evaluate an attack mechanism that adversaries could employ to defeat biometric verification systems which update users templates on a regular basis. While this dissertation focuses on keystroke dynamics- based authentication, we believe that our attack design offers some general insights into how other biometric modalities could be attacked during the template update process. To our knowledge,

this is the first work to investigate such a kind of attack in keystroke biometric authentication systems, and we thus believe that our attack model will motivate a new direction of research seeking to fortify template update schemes against abuse.

3. We propose a residual feature-based attack detection mechanism to identify the fake updates from adversaries. The proposed detection method can protect a biometric authentication system from drifting attacks without compromising the updating performance. A thorough evaluation about how the proposed mechanism protects and affects a biometric authentication system under attacks is presented. Although analysis results are presented in the domain of keystroke dynamics, we believe that our mechanism can be extended to other biometric modalities to prevent adversarial template drifting.

CHAPTER 2

RELATED WORK

2.1 The Aging of Biometric Templates

Template aging refers to the increase in error rates caused by time-related changes in the biometric pattern [30]. Biometric template aging degrades the performance of a biometric system over time, since after a sufficiently long period, the initial enrollment template of a certain subject substantially differs from his current biometric samples, producing lower similarity to the initial template and increasing error rates of the system. In machine learning, the term concept drift is often used to refer to changes in the profile of the data distribution [65]. In biometrics, the drift is often caused by aging, and previous studies have how template aging impacts a biometric system.

Recent studies on human biometric aging traits were used for verification or identification. [29] described a method to model aging variation on human faces based on a statistic face model. The statistic face model was a combination of a shape model and an intensity model, both generated from a list of training face image with 50 parameters. In their work, the aging pattern is represented by an aging function $age = f(b)$ where b is the vector of the 50 parameters, and f is the quadratic function. The function was trained for each individual to fit his/her aging pattern.

They further tested the robustness of a face recognition system with aging simulation. In the training phase, apart from the set of image used to train the face classifier, a second set of image was used to generate the aging function for each individuals in the database. In the test phase, given a face image, an appropriate aging function was used to evaluate the age of the face and the recognition result was generated along with the aging simulation. Their results showed that the improvement of the classification rate was between 5% and 15% with aging simulation. In their following works [27] [28], the focus was the development of artificial age progression algorithms for forensics applications.

[15] analyzed the aging of iris biometrics with a dataset over a three-year period. They compared the match scores distribution for short time-lapse iris image pairs, with an average of one month apart between the enrollment image and testing image, to the match score distribution for iris image pairs with one year, two years and three year time-lapse. A clear evidence of template aging was noticeable with iris images one year apart, with an average of 27% increment of false-reject-rate comparing to the rate with short time-lapse images. Also, the false reject rate increased with increasing time-lapse, and the average increments in false reject rate were 91% and 153% with images of two-year and three-year time-lapse, respectively. In a recent study of biometric aging [47], a multiyear fingerprint dataset was evaluated. Results presented the degrade of system performance without a template updating procedure. A similar result on fingerprint biometric was shown in [64]. While recent works on biometric template aging draw great attention to some modalities, such as face, iris

and fingerprint, there has not been any work on the template aging problem in KD. This dissertation will fill this gap.

2.2 Biometric Template Update

Template update mechanisms keep track of a user's biometric pattern in order to detect and compensate for variations which could degrade system performance. In particular, such mechanisms use the latest user samples accepted by the system to update a user's profile. Two types of template update procedures are studied in the literature: the *supervised* method and the *semi-supervised* method. We describe the two methods in the following paragraphs.

In the *supervised* method input updating data are labeled by a supervisor to ensure only genuine data are updated to user templates. [37] proposed two methods to select user templates in fingerprint biometric, based on multiple enrollments, in order to perform template updates. The first method first clusters the samples then picks the representative sample in each cluster. The other method finds a batch of samples with shorter distances to others. Both template selection and template updates are operated offline, while template updates involves newly accepted genuine samples labeled by the supervisor. An improvement of between 5% and 10% in false accept rate, comparing to the results without template update, was presented with various experimental settings. The *supervised* method is expensive since it requires updating data to be labeled by human expert, or multiple enrollment sessions with attention of users, not mentioning the additional operators to authenticate users in enrollments.

The other option, the *semi-supervised* method, automatically updates user templates with samples collected while the system is operating. In this method, the updating data are selected based on the authentication decision, specifically, only samples accepted by the verification system are eligible to apply template update. [23] proposed a fingerprint verification system with online fingerprint template improvement. Such systems merge the original user template during enrollment with input fingerprints while the system is operating to update user templates. The updating threshold was higher than the verification threshold to limit imposter updates. A recursive algorithm decided the weights of the original template and the update sample while merging. The improvement was presented as a significant reduction of error rates based on the ROC curve, and a better representation in template minutiae.

The applications of template updating mechanism are not limited to the academic area. Some of the recently released electronic devices use template updating procedure to enhance their performance. As an example, both Apple's FaceID [2] and TouchID [1] perform template updating. FaceID updates user template either after a succeeded face authentication, or a rejected face image is followed promptly by a correctly typed backup passcode.

[34] build an experimental analysis to study the template update in face verification. A "self-update" protocol was designed to update user templates with unlabeled data, based on the verification results. With the initial template being set, the rest of the face images are separated into prediction set, unlabeled set, and test set. The prediction set is used to partitions users into different animal groups in "Doddington's Zoo" ([14]). The unlabeled set of each user is combined with the

same amount of images from other users' (imposters) unlabeled set, to operate the "self-update", and the system performance is tested with the test set. The threshold for update is evaluated based on the initial template of each user. With the prediction set, users are cluster into four types of animals: *lamb*, *goat*, *wolf*, and *sheep*, as described in [14].

Lamb: Lambs are users who are vulnerable to imposter attack; their FRRs are higher than others.

Goat: Goats are users who find it difficult to match against their own template. Specifically, goats has higher FAR than other users.

Wolf: Wolves are users who can match well against other users, therefore, they are considered strong attackers.

Sheep: Sheep are users who do not belong to any of the above groups. They are users who generally have good verification performance.

[34] studied the template update effect to each of the four animal groups. Results showed that lambs' templates accepted more updates (65.7% of unlabeled samples were updated) but a high proportion (43.2%) of the updated samples came from imposters. On the other hand, goats had updated lowest amount of updated samples (23.3%) but were less vulnerable to imposter updates (11.8% of imposter updates).

[18] analyzed various template updating approaches in KD, with semi-supervised methods. These approaches update user templates with newly enrolled samples. A threshold was set to select the update samples, and such threshold is stricter than the authentication threshold so that only highly genuine samples are picked. With

two datasets in KD, the experimental results shows an improvement of around 45%. Two ways of updating a user template were used: the *sliding window* and the *growing window*. Both mechanisms are introduced by [24].

Sliding window: A sliding window updating mechanism uses a fixed template size, i.e. the number of samples stored in the template is fixed. Once an eligible updating sample is added to the system, the oldest sample in the template is removed.

Growing window: In this updating mechanism, the template size increases by one sample when an eligible updating sample is added to the template since none of the samples in the template is removed.

A recent study on adaptive keystroke system [66] put forward a keystroke authentication system using template update and "Doddington's Zoo". The system first recognized a user's category according to the animal based categories, then adopted adaptive strategy to remedy problems of the user's class. The *sliding window* and the *growing window* mechanisms were applied, and their system achieved lower than 1% of error rates.

The benefits from using a template update system have been shown in previous studies. Some researchers pointed out that attackers could exploit the template update to intrude a system, by creating fake updates (i.e. [34]) to drift user template. Such adversaries will be described in the next section.

2.3 Adversary Template Drift

A weakness of the template updating scheme is that adversaries can exploit the opportunities of template updates to manipulate user templates. In particular, if an

attacker has some knowledge of a victim's biometric pattern (i.e. snoops a biometric sample from the victim), the attacker could poison the victim's template by adding carefully designed attack samples. As shown in some recent studies [61, 63], avoiding imposter update in an adaptive biometric system could be challenging. [8] investigated the poisoning attack in face biometrics. This attack send a set of fake face images to the camera to gradually compromise a victim's face template. With an image from a victim, the attacker attempts to drift the victim's template towards a targeted template which was decided by the attacker. Each of the attack image was carefully tuned so that each drift was small enough pass the verification. The presented results showed that the attack increased the FAR of the system from 1% to between 5% and 10% with only 5 iterations, and up to 50% with 10 iterations; on the other hand, the genuine users acceptance rate were degraded significantly. The design of the attack requires that the attacker has the perfect knowledge of the verification system (i.e. the verification algorithm, updating policy, victim's templates). [7] further investigate the scenario that the attacker only knows the verification algorithm and the updating policy, with an estimation of victim's template (i.e. a picture of a victim's face from the internet). The results showed that such attack has similar effectiveness to the previous work, with more iterations as a trade-off.

Lovisotto et al. [48] proposed a "backdoor" procedure which allows a face authentication system to accept an attacker's face while minimizing the change of FAR and FRR. The process was accomplished by injecting a series of carefully tuned face images to a victim's template. They claimed that the attack needs minimal knowledge,

but the mechanism was build based on the assumption that the authentication system was a face recognition system using DNN as the verifier.

All works discussed above showed the vulnerability of template updating scheme in biometric verification systems. However, the requirement of the attacker's knowledge to the system (i.e. the verification algorithm, the updating policy) reduces the feasibility of the attack. In this dissertation, a more general template drifting attack is presented in KD, with minimal knowledge and skill requirements from the attacker.

2.4 Defense for Template Drift Attacks

On the defense against such template-drifting attacks, very little research can be found in the literature. The hill-climbing attacks have been reported [50] focusing on brainwave biometric system attacks, which keeps trying different versions of a user's EEG biometric templates until it can eventually access the brainwave biometric authentication system. The synthetic samples were adjusted and improved according to the returned matching scores.

Template protection scheme can be implemented on biometrics system for protecting the template-drifting attacks through cryptographic protocols [49, 51]. Gomez *et al.* [45] used the uniform score quantization to enhance the system security, but it actually restricted the system's template updating ability by setting the number of the desired quantization levels. Giot *et al.* [19] tested a "two-threshold" updating strategy in order to filter the attack samples. In particular, samples that were added to the template need to satisfy an "updating threshold" which is stricter than the

authentication threshold. Therefore, only the "highly genuine" samples were updated. In this test, a mixed pool of genuine and imposter input samples were used to operate the template updates. The amount of genuine sample was fixed, and the rate of amount of imposter samples in the pool was decided by a parameter. In the enrollment phase, the system created two templates which were the same for each user. In the verification phase, each input sample compared with each of the two templates to generate verification scores. The final score was the average of the two scores. An authentication threshold decided whether the sample was accepted, and an updating threshold further decided whether to use the sample to update user templates, if the sample passed the authentication. The authentication threshold was the EER based on the training, and two sets of updating threshold were used: 1) the EER threshold which was the same as the authentication threshold, and 2) a threshold at 1% FAR. The experimental results showed that a stricter threshold significantly reduced the imposter update rate from around 10% to lower than 0.2%. On the other hand, the genuine update miss rate was increased from around 20% to over 80%. These results clearly showed that the "two-threshold" strategy limited the effect of imposter updates by reducing numbers of imposter updates significantly. However, such mechanisms may also affect the updating performance of the system since some genuine samples were also rejected for updating.

Different from all prior efforts, in this dissertation we will propose a mechanism to defend "poisoning" type of attacks without sacrificing the updating performance.

CHAPTER 3

ANALYSIS OF BIOMETRIC TEMPLATE EVOLUTION

In this chapter we study the evolution (or aging) of keystroke biometrics features over time. We use a range of pattern analysis and statistical methods for this analysis. Observations made in this section form the basis for the attack designs and performance evaluations to be undertaken in the proceeding sections. We first describe the dataset used for these evaluations before giving details of the results of our analysis.

3.1 Dataset Description

In this dissertation we use two keystroke dynamic datasets. One collected by Carnegie Mellon University (CMU data), the other one collected by Louisiana Tech University (LTU data). Descriptions follow.

CMU data: This is a public dataset with keystroke typing data for 51 users. All data was collected on a laptop Windows computer with an external keyboard. The data collection was splitted into 8 sessions, and users were allowed to complete only one session each day. The time span of 8 sessions of data collection varies from one week to one month. In each session, users were asked to type a 10-character phrase .tie5Roanl' for 50 times. A total of 400 typing repetitions were collected from each

user. A Windows application was developed for the data collection. If a mistype happens, users are prompt to type the phrase over again.

LTU data: The LTU data was collected in three sessions over two years, which are fall 2009, fall 2010 and fall 2011. Totally 138 users joined all 3 sessions of data collection. In each session, users are asked to type the same 58-character phrase multiple times. The phrase was I am an undergraduate student of Louisiana Tech University. Twelve repetitions of the phrase were collected in the first session (fall 2009), and 15 repetitions were collected in each of the second and the third session (fall 2010 and fall 2011). All keystroke data was collected on a Windows PC with a physical keyboard. A Windows application was developed to guide users to type. The application displays a window with a line of text showing the word phrase, and a text box below the line of text for users to type. Users need to correctly type the whole phrase and hit Enter button to type the next repetition. If any mistype happens, a single click of Backspace button will clear everything in the text box, and users need to type the phrase over again. The keypress time and key release time of each of the 58 characters (including 8 space characters) were recorded by the application, and 58 KHT and 57 KIT were generated for each typing phrase.

3.2 Keystroke Verification Algorithms

Killourhy et al. [26] compared the performances of 14 keystroke verification algorithms. They collected keystroke time data for typing password from 51 users. Each user had 8 session of data with 50 repetitions in each session. The first 4 sessions of data from each user were used to build the user template in the training phase,

and the last 4 sessions built the testing set for both genuine test (200 repetitions of genuine samples) and imposter test (keystroke samples from other 50 users). With each keystroke verification algorithm, the anomaly score of testing samples are calculated, and a comparison of 14 algorithms was generated based on the error rates. three verifiers stood out to be best, which are: 1) Scaled Manhattan, 2) Outlier Count, and 3) Nearest Neighbor. We apply the first verifier, Scaled Manhattan in this dissertation. The second algorithm Relative is another well-known high-performance KD verification algorithm that was first proposed by Gunetti et al. [3]. The latter algorithm was particularly used to complement the first three algorithms because they are all Euclidean distance-based. Details of the verification algorithms follow:

Scaled Manhattan Verifier (SM): This detector was described by Araujo et al. [5] The training phase involves computing the mean and mean absolute deviation of each feature in a user template, while the mean absolute deviation is calculated with the following equation:

$$a_i = \sum_{j=1}^{n} |\mu_i - x_{ij}|/n \tag{3.1}$$

μ_i is the mean of the i^{th} feature in the user template, x_{ij} is the i^{th} feature value of the j^{th} sample in the template and n is the number of samples in the template. The test phase involves computation of the anomaly score using the expression:

$$s = \sum_{i=1}^{m} |y_i - \mu_i|/a_i \tag{3.2}$$

y_i is the i^{th} features of the test vector with m number of features.

Relative Verifier (R): This verifier was described by [Bergadano et al., 2002]. In the training phase, the verifier computes the mean of each feature from a user template. In the test phase the verifier assigns ranks to each of the elements in the test and mean vectors, before computing the degree of disorder between the rank vectors. The normalized degree of disorder equals the anomaly score.

Fusion Verifier (F): To further enhance the rigor of our performance analysis, we experimented the fused verifier, since verifier fusion is known to improve the performance of biometric classifiers [33, 34]. Note that our choice of the SM and R verifiers for the fusion classifier was because the SM verifier was the best performing verifier in the earlier mentioned study [26], while the R verifier, by virtue of being the non-Euclidean verifier, should capture the users' typing traits from a perspective that complements that of the SM verifier.

Before fusing scores, we perform a genuine score centric normalization, in which each score s is normalized to obtain a new score s' , where s' is calculated as:

$$s' = (s\mu G)/\sigma G \tag{3.3}$$

μG and σG are respectively the mean and standard deviation of the genuine training scores. For all fusion we use the weighted sum-rule [11] with equal weights.

3.3 Analysis Results

In the following subsections we present results on different perspectives of feature evolution over time. We first compare the distribution of feature changes (Section 3.3.1). Details of the feature distributions were already introduced in Section

3.3.1. Following the distribution comparisons we perform statistical significance tests to assess the significance of feature variations seen in each individual feature and different sizes of feature vectors (Section 3.3.2). In the last part of this analysis, we perform user classification using training and testing data collected at different points in time to determine whether or to what extent the feature changes seen via the significance tests translate into decrements in user classification accuracy.

3.3.1 Distribution of Feature Changes

Figures 3.1 are the CDFs of the absolute changes in users' mean values for each feature between 2009 and 2011. For an arbitrary user who participated in our data collection experiments in both 2009 and 2010, the change in mean feature values seen between 2009 and 2010 is computed as follows. Over the 12 instances provided in 2009, we compute the mean of each KHT and each KIT. Since there were 58 distinct KITs and 57 distinct KHTs in the paraphrase used for our experiments, this process creates a 57-dimensional vector and a 58-dimensional vector for each user. Over the 15 instances provided in 2010, the KHT mean vector and the KIT mean vector are computed in the same way for each user. Finally, for each user a differences vector is computed by subtracting the KHT mean vector computed from the 2009 dataset from that computed for the 2011 dataset, with the same process repeated for the KIT mean vector. The difference is computed as an absolute value. Figures 3.1 shows the cumulative distribution of the values in the differences vectors over the population for the KHTs while Figure shows a similar distribution for the KITs.

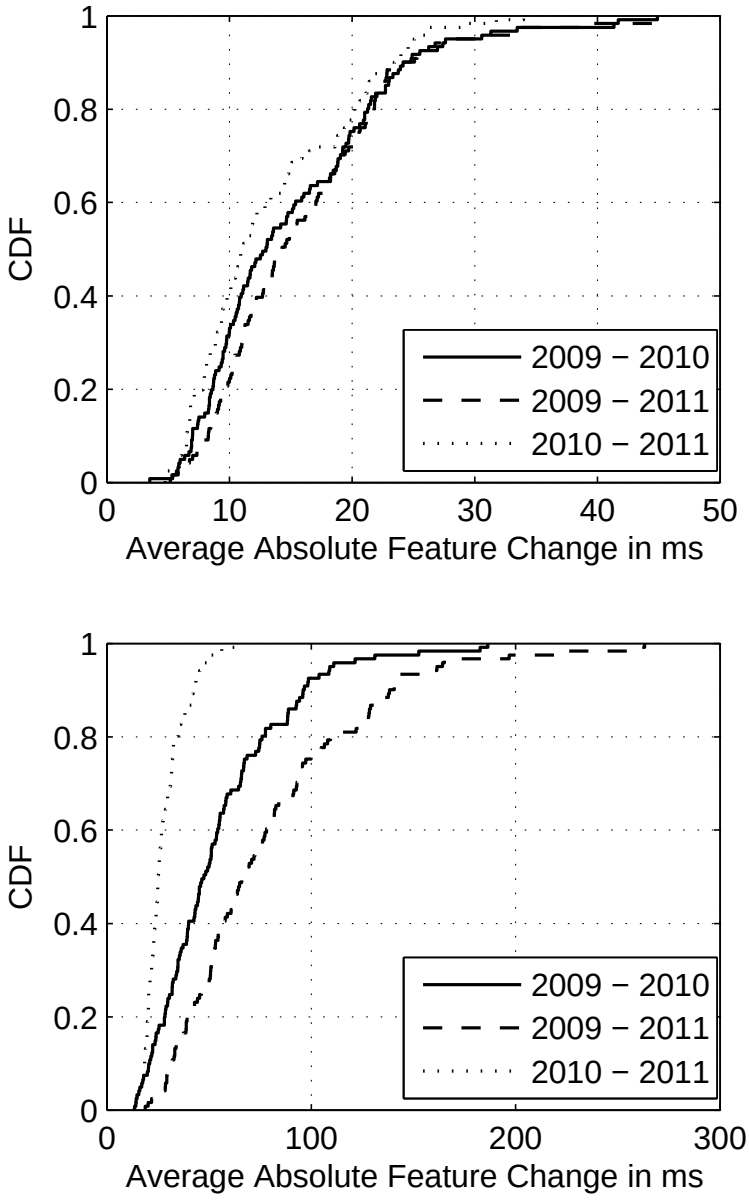


Figure 3.1: CDF of User Feature Change in Milliseconds (ms) Across 3 Phases.

These distributions provide insights into the changes seen in the different features on average. Observe that the KITs depict much higher differences than the KHTs; the differences seen with the KITs go as high as 200+ ms while those seen with the KHTs only reach $40ms$ at most. This points to KITs being more unstable than the KHTs over time. The figures further show that there were some changes that were considerably higher than the mean change behavior. E.g., over 20% of the differences in KHTs were more than $20ms$ while over 20% of the KITs were more than $80ms$. From this graph it is not possible to determine whether these considerably higher changes were due to user behavior or due to the identities of the keys typed. However, the figures indicate the existence of a change, which we further analyzed in terms of EERs and with the aid of statistical tests of significance to get a concrete picture of these changes in user features.

3.3.2 Analysis with Tests of Significance

In the previous plots (Figure 3.1) we studied the evolution of each individual feature in a user's feature vector. Here, we study evolution from the perspective of an entire feature vector. We did this in two ways:

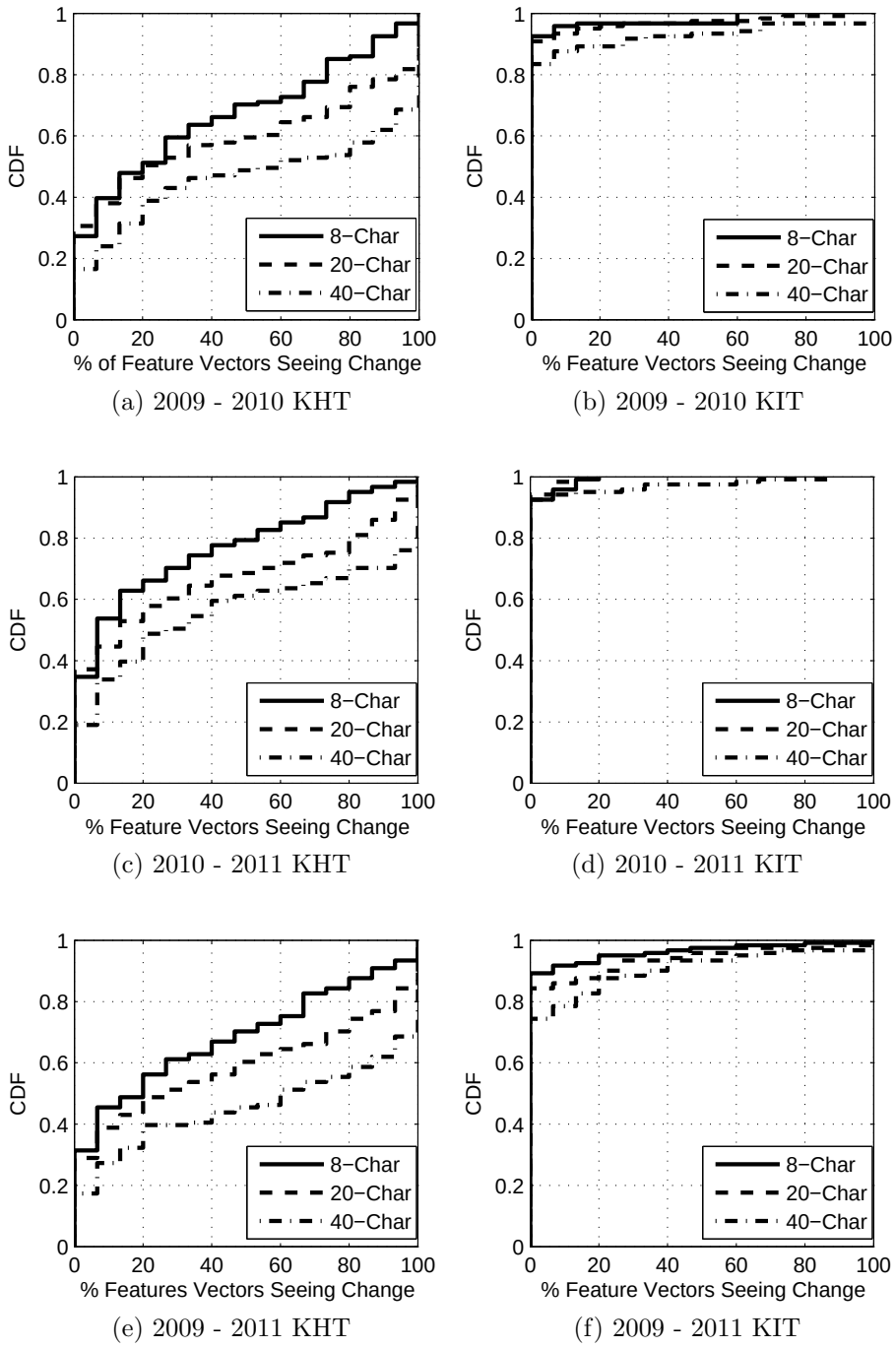
Approach 1: In the first approach, we use the typing samples collected in 2009 (or 2010) to compute a vector of feature means for each user, and then compare this vector with a similarly created feature-means vector built for the same user from typing samples collected in a later year.

For many keystroke verification algorithms (e.g., see [26] for a survey), this vector is the main building block of a user's profile, and we thus believe it should

be a plausible representation of a user's typing pattern at any given point in time. In the rest of this paper, we refer to this vector as the user's profile. For a range of string lengths between 8 and 58 characters, we run the K-S test for each user, the null hypothesis being that the particular user's reference profile for the string in question does not vary significantly from the profile built using data collected in a later year. These tests will give some answers as to whether a user could take up different identities (in keystroke terms), if their profile was to be built at different points in time.

Approach 2: In our second approach to studying keystroke feature vector evolution, we test if each of the individual samples (feature vectors) provided by a user in 2010/2011 differed significantly from the particular user's reference profile. Unlike the previous approach where the comparison between users' profiles gave a consolidated view of the evolution of a user's keystroke traits, these tests will give some insights into how well individual authentication attempts match with a profile which was built years before the attempts are made. For the same range of string lengths used in the previous tests, we run the tests using the feature vectors provided by each user in the later years (2010 or 2011).

Thus while the first method uses one test per user per string length, the second approach uses several tests per user per string (the number of tests depends on the number of feature vectors provided by the user in the later year), the null hypothesis in each case being that the feature vector under consideration does not significantly differ from the user's reference profile.



(a) 2009 - 2010 KHT

(b) 2009 - 2010 KIT

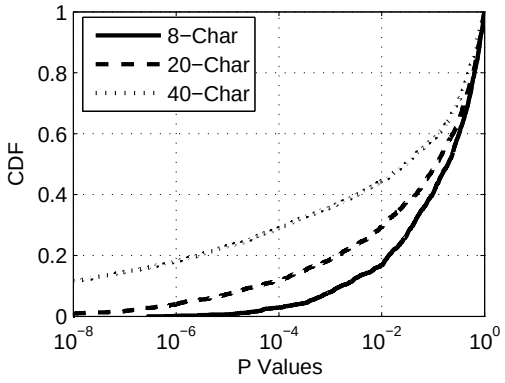
(c) 2010 - 2011 KHT

(d) 2010 - 2011 KIT

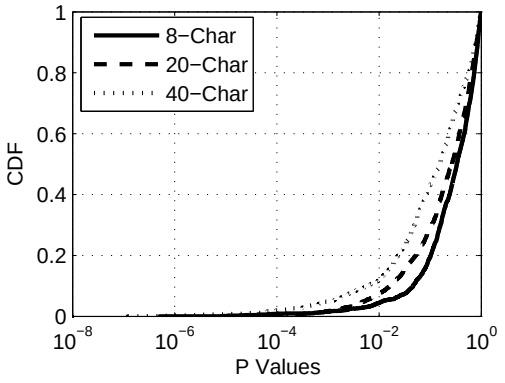
(e) 2009 - 2011 KHT

(f) 2009 - 2011 KIT

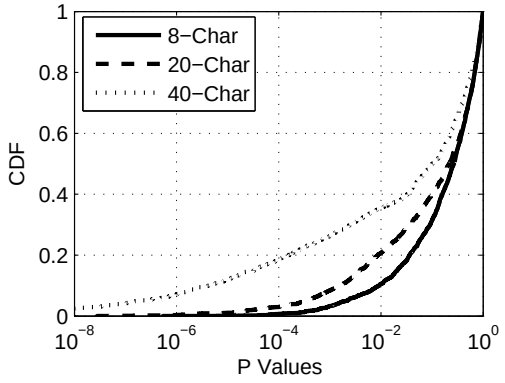
Figure 3.2: Distribution of the Percentage of Users' Samples Significantly different from an earlier profile.



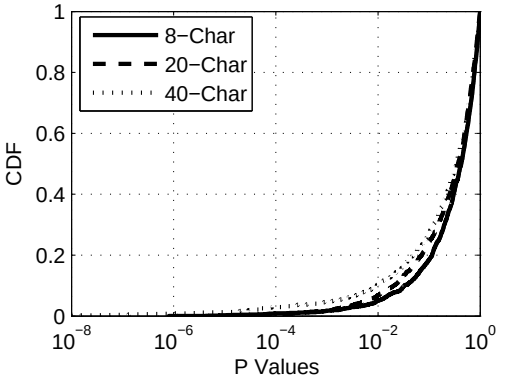
(a) 2009 - 2010 KHT



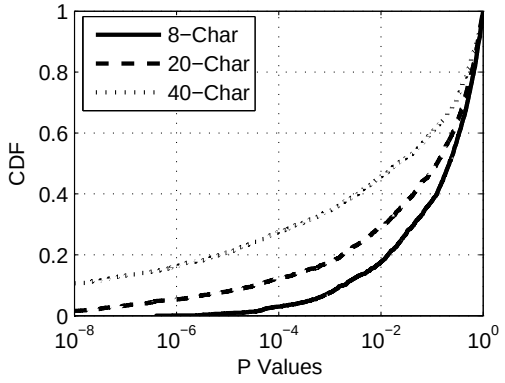
(b) 2009 - 2010 KIT



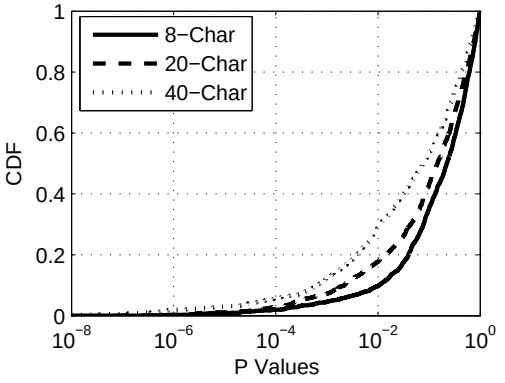
(c) 2010 - 2011 KHT



(d) 2010 - 2011 KIT

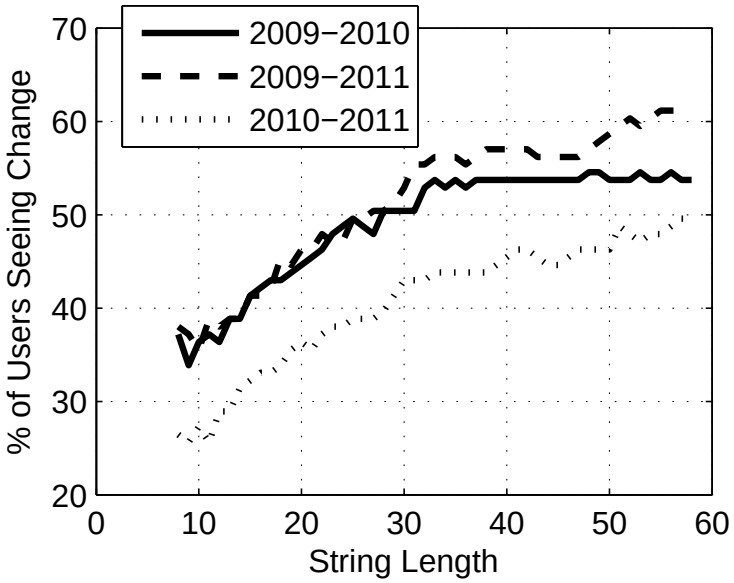


(e) 2009 - 2011 KHT

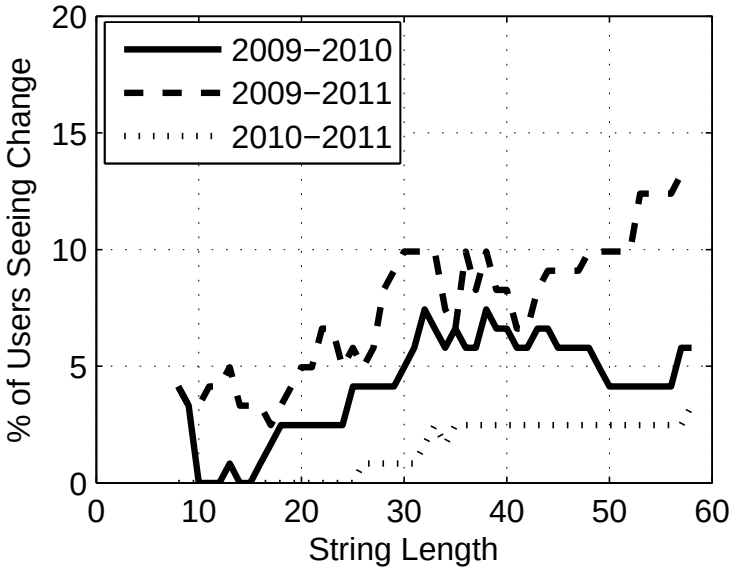


(f) 2009 - 2011 KIT

Figure 3.3: Distribution of the P values from the significant tests.



(a) KHT



(b) KIT

Figure 3.4: Percentage of Users for whom the profile significantly different from earlier profile with various string length.

Figures 3.2, 3.3 and 3.4 summarizes our findings from these feature evolution tests. In Figure 4-2, we plot the CDF of the percentage of the feature vectors (from the 2010/2011 experiments) that depict a significant change when compared to the respective users' reference profiles. The reference profiles are either based on the 2009 dataset or the 2010 dataset. For example, the plot labeled 2009-2010 refers to a reference profile which was built based on 2009 dataset while the profile used for comparison was based on data collected in 2010.

Observe that for both KHTs and KITs, the percentage of feature vectors seeing significant change increased with increasing string length. The increments seen with the KITs are more subtle but still apparent from the plots. The figures generally confirm that the longer strings are more likely to be associated with unstable keystroke profiles than the shorter strings (See Figure 3.3 for the distribution of P values returned from the hypothesis tests used to plot Figure DD. The critical value used was 0.05. P values lower than 0.05 indicate rejection of the earlier described null hypothesis.)

Figure 3.4 shows the results for the experiments run following **Approach 1:**. The figure confirms the earlier observed trend, as the percentage of 2010/2011 user-profiles depicting significant variation from the respective users' reference profiles generally increases with increase in string length.

While it has previously been shown that aging of users' features can degrade the performance of a keystroke verifier [Killourhy and Maxion 2010], no previous work has investigated how keystroke features age with time. Our results here provide empirical evidence detailing the nature of keystroke template evolution over a long period.

We note that the discussion in this section applies to the evolution of keystroke features from the statistical perspective and does not necessarily point to how/whether feature evolution may result into authentication failure for the user in question. Next, we address how the analyzed keystroke feature evolution patterns impact the classifier Equal Error Rates.

3.3.3 Analysis of Equal Error Rate (EER) Performance

The performance of a keystroke verification algorithm is often measured with error rates, specifically, false accept rate (FAR), false reject rate (FRR), and equal error rate (EER). FAR is the probability that a biometric verification system incorrectly accept a verification attempt by an unauthorized user. The FAR typically is calculated as the number of false acceptance divided by the number of access attempts by imposters. FRR is the probability that a system incorrectly rejects an access attempt from a genuine user. FRR is calculated as the number of false rejects divided by the number of verification attempts by genuine users. EER is the rate at which the FAR and the FRR are equal. EER can be obtained by the ROC curve, as in Figure 3.5.

To measure the performance of the keystroke verification algorithms, we generate the anomaly score of each testing sample, and the FAR and FRR depend on how the threshold of the anomaly score is chosen. The choice of threshold establishes the operating point of the verifier on the ROC curve. Over the continuum of possible thresholds, the ROC curve illustrates the FAR and FRR that would be attained at each possible detector operating point. EER is then calculated as in the intersection of the ROC curve and the line $FAR = FRR$.

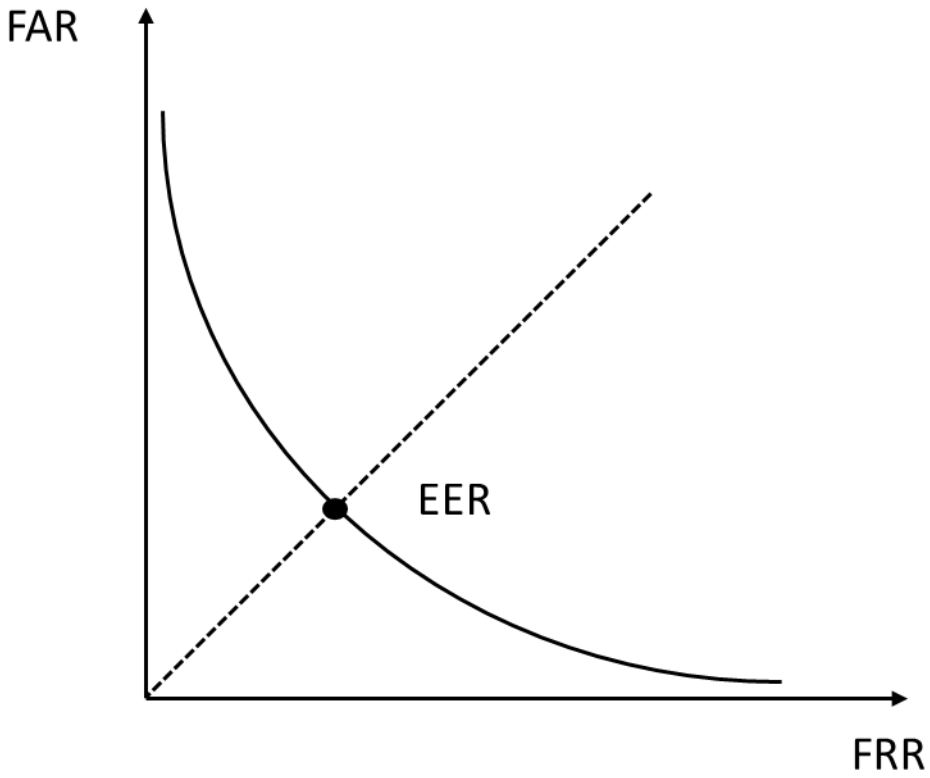
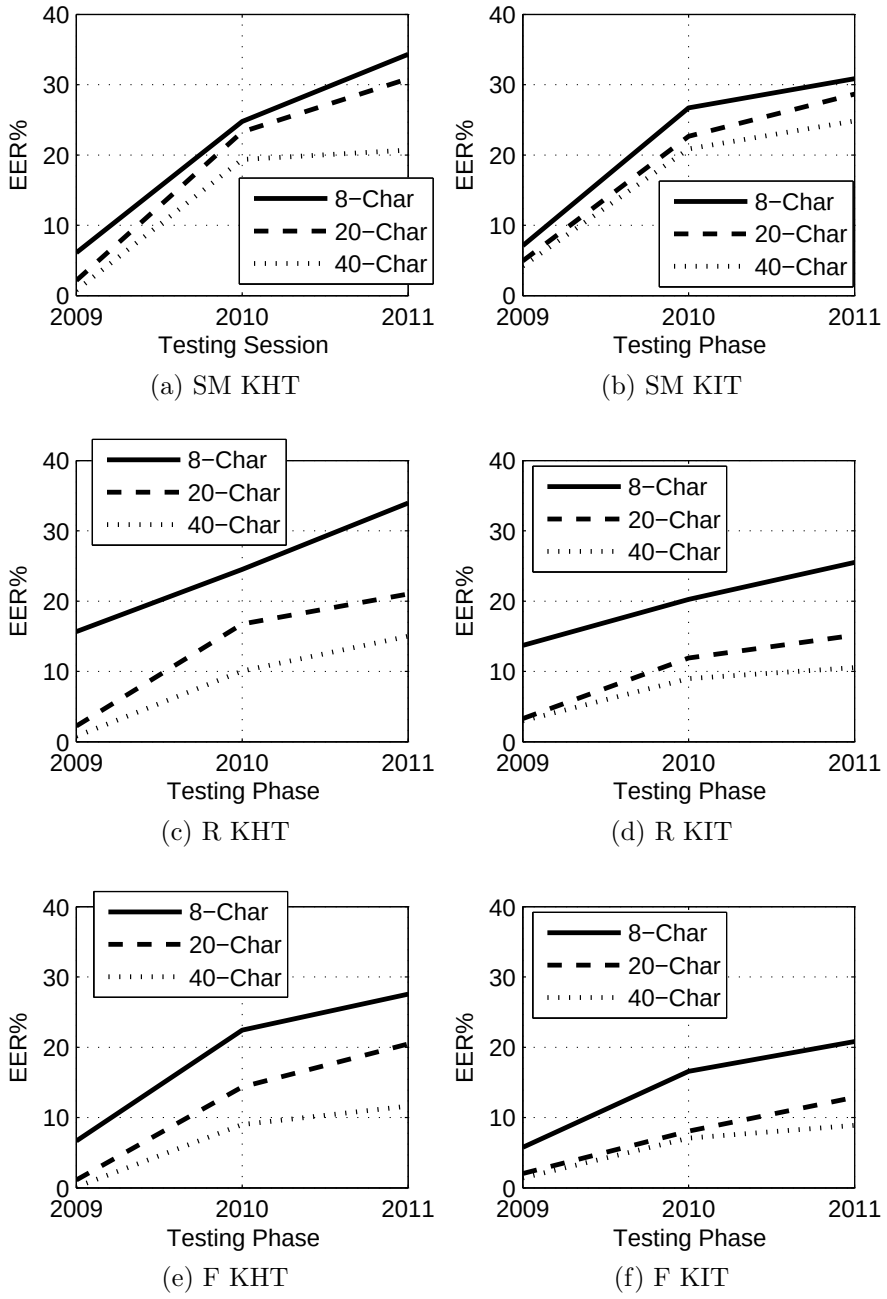


Figure 3.5: An example ROC curve. The curve shows the trade-off between the false accept rate (FAR) and the false reject rate (FRR). The intersection is the equal error rate (EER).

Figure 3.6 shows the impact of template aging on classifier performance. Three classification algorithms have been used for this analysis, i.e., Scaled Manhattan algorithm (SM) [], R verifier (R) [] and a fusion of the R and SM verifiers based on the weighted sum rule (F)[].

For each algorithm we perform training and testing in three different scenarios:

1. Data collected from each user in 2009 is divided into two portions; one of which is used for training and the other for testing. 5 samples typed in 2009 are augmented with 15 samples generated by a Monte Carlo simulation to create a training set of 20 samples. Testing is based on 7 samples collected in 2009.



(a) SM KHT (b) SM KIT

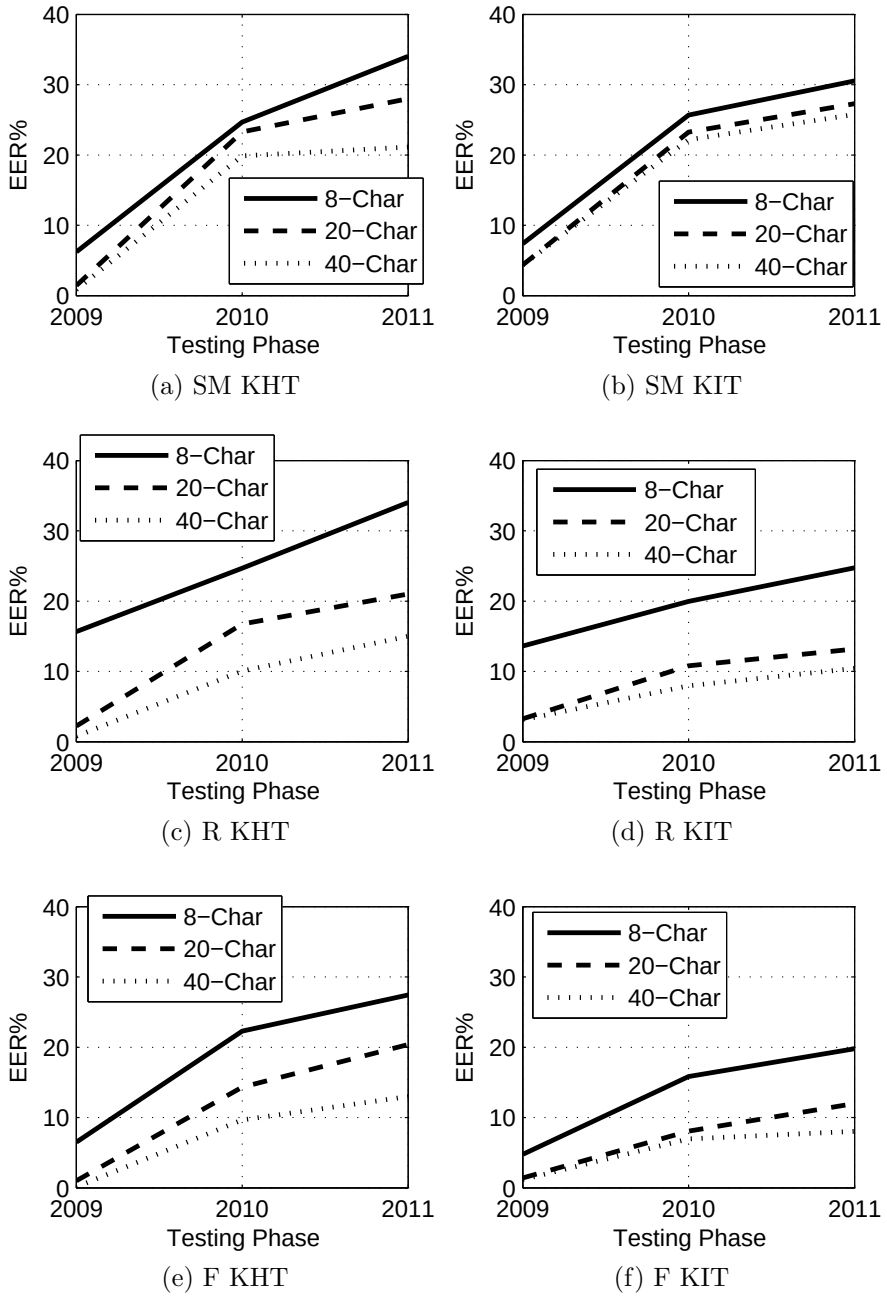
(c) R KHT (d) R KIT

(e) F KHT (f) F KIT

Figure 3.6: EER performance with different testing phase. 20 training samples are generated with MC.

2. Training is done similarly to scenario 1 while all 15 samples collected in 2010 are used for testing.
3. Training is done similarly to scenario 1 while all 15 samples collected in 2011 are used for testing.

Figure 3.6 (a), (c) and (e) show the performance of the classifiers when training and testing was done based on KHTs respectively extracted from an 8-character string, a 20-character string and a 40-character string. The figures show that for all 3 verifiers, the classifier Equal Error Rates were lowest for scenario 1, followed by scenario 2 and highest for scenario 3. The same trend is seen when the features used were KITs (see Figure 4-5 (b), (d) and (f)), and when the Monte Carlo simulation was used to produce 50 samples (see Figure 3.7) as opposed to 20 samples. The reason behind this trend is that as time went by, each user's features underwent a drift, which resulted in worsening classification performance over time. In all cases the fusion verifier performs at least as well as the individual verifiers, however, it still depicts the same trend. These results provide solid evidence for the evolution of users' templates over time, and the need for template update mechanisms to remedy this situation.



(a) SM KHT (b) SM KIT

(c) R KHT (d) R KIT

(e) F KHT (f) F KIT

Figure 3.7: EER performance with different testing phase. 50 training samples are generated with MC.

CHAPTER 4

DESIGN AND PERFORMANCE EVALUATION OF THE FROG BOILING ATTACK

4.1 Design of Attack

The previous section has shown evidence for the evolution of keystroke features over time. A standard solution to this kind of problem is the usage of template update mechanisms. Such mechanisms make corrective changes to a user's template when the user's features are deemed to have changed significantly relative to the stored template. While ours is the first work to provide concrete insights into the evolution of user's keystroke features over time, there are interestingly already some existing proposals on corrective mechanisms (or template update mechanisms) against template aging. In this section we investigate the notion that these corrective mechanisms can be exploited to inject malicious samples into the template. Specifically, we design an attack called a frog-boiling attack that seeks to stealthily drift a user's features. We evaluate the performance and practicality of the attack in different settings.

Rubinstein *et al.* [43] introduced the term "Boiling Frog" as a type of poisoning attacks in which "the adversary slowly, but increasingly, poisons the principal components by adding small amounts of chaff, in gradually increasing quantities." Chan-Tin *et al.*[9] coined the term Frog-Boiling attack in reference to an attack which

uses a sequence of carefully tuned fake updates to stealthily perturb the coordinate system of a network. This stealthy modification of the coordinates is likened to the tale of a frog which ended up being boiled in water just because the temperature was increased in very small undetectable steps. In this dissertation, we adapt the philosophy of the Frog-Boiling attack to the domain of keystroke based verification. In particular, our frog-boiling attack is built upon the following assumptions:

1. an adversary who is able to snoop on a user's typing session and use the captured timing data to synthesize and replay authentication attempts with the aid of software tools (i.e. key-loggers and keystroke generators); and
2. a keystroke verification system that uses template updates to correct variations seen in a user's template over time (i.e., template aging).

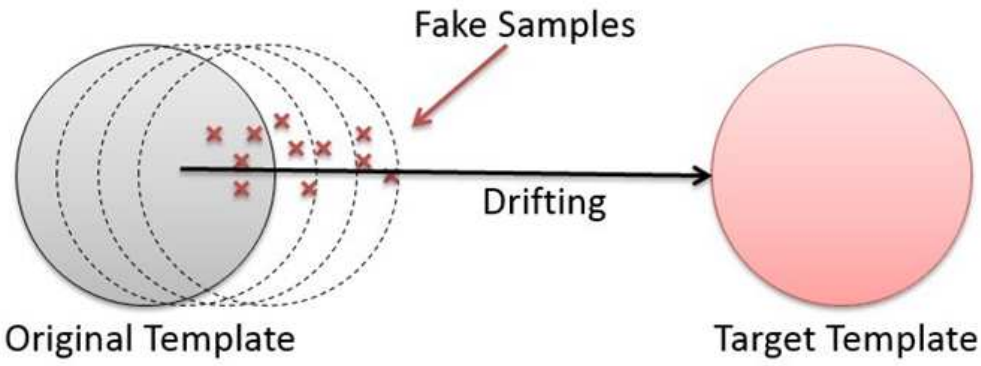


Figure 4.1: Idea of the Frog-Boiling attack. Attacker intends to drift a user template (gray circle on the left) to a target, by inserting fake samples (red crosses) to a user template.

Figure 4.1 presents the general idea of the Frog-Boiling attack. The gray circle on the left represents the original template of a KD user. The Frog-Boiling attacker seeks to drift the user template towards some target template, using carefully tuned fake samples which marginally deviate from the user's own samples. This marginal

variation from the user's own template helps fool' the system into accepting them, with the cumulative impact of these samples eventually aimed to transform the user's template into a target template that is very different from that of the genuine user. An example of the target template could be a template that belongs to the attacker (that would eventually give the attacker unfettered access to the system), or some generic weak template aimed to degrade the authentication performance of a user.

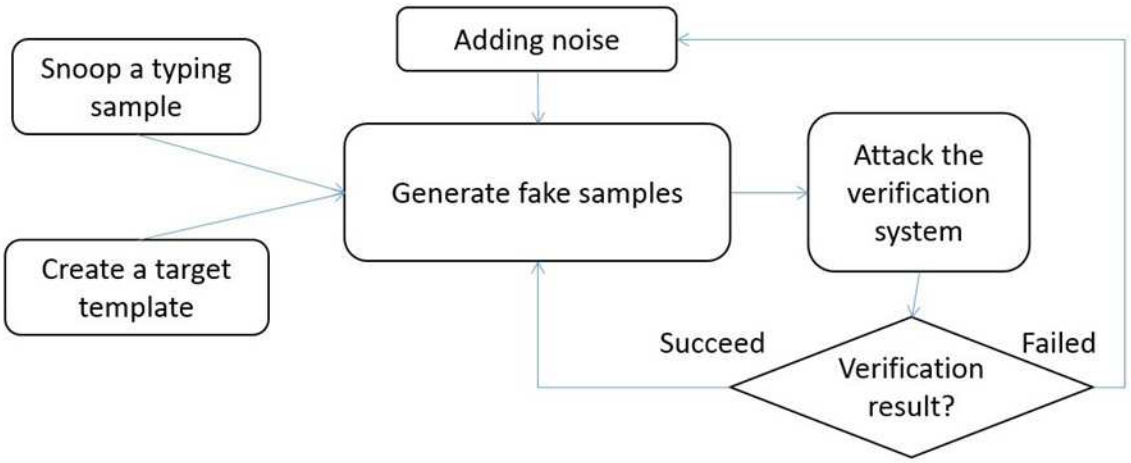


Figure 4.2: Idea of the Frog-Boiling attack. Attacker intends to drift a user template (gray circle on the left) to a target, by inserting fake samples (red crosses) to a user template.

Figure 4.2 is a flow chart of the Frog-Boiling attack. A list of fake samples is generated based on a snooped user sample and a target template chosen by the attacker. These samples are well-tuned to create a drift to a victim's template, while staying in the range to be verified by the system. Gaussian random noise is added to the samples to create some randomness, hence avoiding suspicion. Algorithm 1 shows a formal view of the attack.

Input: $F, \bar{D}, N, n$
Input: $\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_n\}$
$\Delta \leftarrow (\bar{D} - F)/N$;
for $i \leftarrow$ *1* **to** P **do**
  //P iterations in total
  for $j \leftarrow$ *1* **to** n **do**
    $x \leftarrow \mathcal{N}(1, \sigma_j)$;
    if $i \leq N$ **then**
      $F' \leftarrow (F + (i-1) \cdot \Delta) \cdot x$;
    end
    else
      $F' \leftarrow (F + N \cdot \Delta) \cdot x$;
    end
    //attack vector
    if Verification$(F') == true$ **then**
      //verifier accepts vector F'
      $v \leftarrow 1$; **break**();
    end
  end
  if $v \neq 1$ **then**
    break(); //Abort attack
  end
end

ALGORITHM 1: The Frog-Boiling Attack

Given a user with a keystroke template denoted by S, the Frog-Boiling attack (see Algorithm 1) seeks to transform S towards a *destination* template D, where D stands for an arbitrary keystroke template, which could, for instance, be built from the attacker's own keystroke features, or from data collected from several users over a population. We shall refer to the latter form of the *destination* template D, as a *Population Template*, and the former as a *User-specific Template*.

$$F' = \begin{cases} x \cdot (F + (i-1) \cdot \Delta), & 1 \leq i \leq N \\ x \cdot (F + N \cdot \Delta), & i > N \end{cases} \tag{4.1}$$

Equation 4.1 generalizes the feature vector F', used to make the i^{th} Frog-Boiling attempt. F represents the feature vector that the adversary directly synthesizes from data collected during the initial keylogging process (see Assumption 1), while $\Delta = (\bar{D} - F)/N$ is the vector of small latency modifications being used to slowly drift S towards D with the aid of the *Sliding Window* template update mechanism. $\bar{D}$ is a vector whose elements are obtained by computing the mean values of the different features in the template D, while the scalar N represents an arbitrary number of small steps used during the first stage of the template transformation (i.e., the stage represented by the interval $1 \leq i \leq N$ in Equation 4.1).

The noise term x is added to make each forged feature value appear random, since a set of regularly spaced feature values could easily raise suspicion. We model the noise using the Gaussian random variable with fixed mean, 1, and standard deviation $\sigma_j \in \{\sigma_1, \sigma_2, \cdots, \sigma_n\}$, where $\sigma_1 > \sigma_2 > \cdots > \sigma_n$. The default value of σ_j is set such that $\sigma_j = \sigma_1$, the lower values of σ_j being used only when the forged feature vector F' fails to authenticate against the victim's template.

In these cases of failed authentication, where the forged feature vector F' fails to authenticate against the victim's template, the Frog-Boiling attack stipulates a total of n re-authentication attempts, for which the standard deviation of the noise term is reduced after each failed authentication. Reduction of the noise term is done through use of a lower standard deviation for the Gaussian variable X. If authentication still fails after the n times, the attack against that particular user is aborted. For the attacks in this work we used $n = 5$, since many password authentication systems permit an average of 5 failed authentication attempts before initiating additional

defensive measures. Details of the other attack parameters are discussed in Section 4.3.

4.2 Baseline Evaluation

Before evaluating the attack, we performed a set of baseline experiments whose results will serve as a reference point to gauge the impact of the Frog-Boiling attack. These experiments were based on data collected from the first 46 of the 51 users, as samples collected from the remaining 5 users were reserved for the design of the Frog-Boiling attack. The steps followed for each user in the training and testing process of the baseline experiments are listed below:

1. With one of the first 46 users designated as a genuine user, the other 45 users are designated as the impostors. Data collected from the first 2 typing sessions of the genuine user (equivalent to 100 typing repetitions) is then used to train each of the six verifiers.
2. To generate genuine scores of a given genuine user, data from Sessions 3 (equivalent to 50 typing repetitions) is used to attempt the user's model.
3. Finally to generate impostor scores, data from the first five typing attempts of each impostor in Sessions 3 is used to attack the genuine user's model.

Table 4.1: The mean (μ_{EER}) and standard deviation (σ_{EER}) of the EER of each of the 3 verifiers at baseline. EERs are expressed as a percentage.

	SM	**R**	**Fusion**
μ_{EER}	13.29	18.10	9.97
σ_{EER}	7.73	9.20	6.10

Table 4.1 summarizes the performance of the classifiers in terms of their EER at baseline. The fusion verifier performs the best, and SM verifier performs better among the two single verifiers. The next section details the attack process, and how the baseline performances of the different verifiers are affected by the attack.

4.3 Attack Settings

We perform two types of Frog-Boiling attack: the *user-specific attack*, for which the target template is a specific attacker, and the *population attack*, for which the target template is the average of a population. For both biometric systems, the last five users in each dataset are designated as the attackers, and the Population template is computed based upon the profiles of the last 5 users. The snooped samples used to initiate the attack are obtained from Session 4 of the keystroke victim.

The standard deviation of the noise distribution is drawn from the set $\sigma = [0.4, 0.3, 0.2, 0.15, 0.1]$, while $N = 100$ and $P = 300$ (see Algorithm 1, for meanings of parameters). The large value of N ensures that the individual template drifts causing the transition from S to D are sufficiently small, while the large value of P enables the victim's template to eventually get completely flushed by samples from the Frog-Boiling attack attempts made when the counter variable, i, is such that $i > N$ (see Equation 4.1). We set $N = 100$ and $P = 200$.

We perform the Frog-Boiling attack in two scenarios, the so-called *pure attack* and *mixed attack*. With pure attacks, all authentication attempts during the Frog-Boiling attacks are attack samples, hence the attacks proceed while a victim is not using the system. The results of the pure attacks will show the full potential of the

Frog-Boiling attack. However, a large amount of attack attempts during a short amount of time could be suspicious (considering a victim may still use the system occasionally). If the attack could proceed without interrupting a victims normal usage, the intrusion would be more difficult to detect. Therefore, we also investigate the more challenging and demanding *mixed attack*, in which the biometric system receives both genuine and attack attempts. A prior study [36] used the Poisson distribution to model users' login behaviors, and we use a similar approach in this work.

In our experiments, Poisson random numbers that represent the instant at which a login attempt is made are generated for both genuine and attack attempts using Equation 4.2, with $\lambda = 2500$. n represents the number of attempts generated. In our experiment, $n_G = 300$ for genuine samples, $n_A = 150, 300,$ or 600 for attack samples, which correspond to the ratios between the genuine and the imposter attempts as 2:1, 1:1, or 1:2, respectively. With *pure attack* we flush victims' template with 200 attack samples. With *mixed attack*, we use a virtual timeline to show the process of the attack. All the genuine and attack attempts are Poisson distributed on the timeline, and the process of the attack lasts 200 time units with keystroke system. The expected numbers of genuine attempts (through the Poisson random generator) within the timeline are 200, and the expected numbers of attack attempts varies based on the mixed rate. Note that even we use fixed rates on the amount of genuine and attack samples, since all the attempts are randomly generated, there could be consecutive genuine or attack attempts.

$$[x_1, x_2, \cdots, x_n] = \mathcal{P}(\lambda) \tag{4.2}$$

Table 4.2: The mean and standard deviation of the EERs for the three verifiers after the Frog-Boiling attack. EERs are expressed as percentage.

Attacker	**Verifier**		
	SM	**R**	**Fusion**
Attacker #1	64.27 (23.51)	37.43 (34.46)	47.43 (40.79)
Attacker #2	62.28 (21.98)	30.48 (27.44)	47.52 (36.04)
Attacker #3	56.77 (33.19)	20.65 (29.67)	24.00 (31.78)
Attacker #4	52.37 (25.93)	16.09 (17.94)	22.83 (26.88)
Attacker #5	53.85 (21.96)	23.04 (25.65)	32.11 (30.71)
Population attack	54.80 (23.98)	20.59 (10.83)	26.62 (13.07)

4.4 Attack Results

Table 4.2 summarizes the performance of the three verifiers after the full Frog-Boiling attack was launched. To compute the EERs, we subject the victim's transformed template to the same training and genuine-score generation procedure that was used in Section 4.2, but generate the impostor scores using the attackers' samples from session 2 (50 repetitions). Impostor samples are perturbed with noise before the attack is launched. A total of 1000 impostor attempts are made.

Table 4.2 generally reveals that all classifiers see increased EERs as a result of the attack. The EER increments vary across attackers, since, for instance, attacker #1 causes EER increments as high as 400% (e.g., SM verifier, from 13.29 (see Table 4.1) to 64.27 (Table 4.2), Fusion verifier, from 9.97 (Table 4.1) to 47.43 (Table 4.2), etc.), while attacker #5 affects all verifiers just slightly. This trait is a direct result of our attack model, in which the victim's *destination* template directly depends on the attacker's template. As such, an attacker with a weak *destination* template should, with high likelihood cause higher EER increments than an attacker with a strong template.

Another interesting observation about the results in Table 4.2 is that the R verifier, which performed worst at baseline, appears to resist the attack more than the other verifiers. This could be because the R verifier's mechanism rotates around the relative ranks of the features (see Section 3.2 for its description), yet the SM specifically focus on the magnitudes of the feature values, which are directly modified by the Frog-boiling attack.

The relatively good performance of the *A-fusion* verifier could also be attributed to the resistance of the *R* verifier (which is one of the individual verifiers used in the fusion) to the attack. Meanwhile, the significant performance degradation seen by the *S-fusion* verifier was likely because the set of well-performing users that are not considered for fusion by this verifier could have changed into weak users after the Frog-Boiling attack. This observation suggests that in an adversarial environment, *S-fusion* verifiers would have to be dynamic, continuously monitoring users' performance after enrollment. The implication of the S-fusion verifier's behavior under attack is that for biometrics modalities where a template update scheme could be abused, user's the distinction between good and poor users would have to be continuously done, as opposed to biometric performance would have to be continuously monitored.

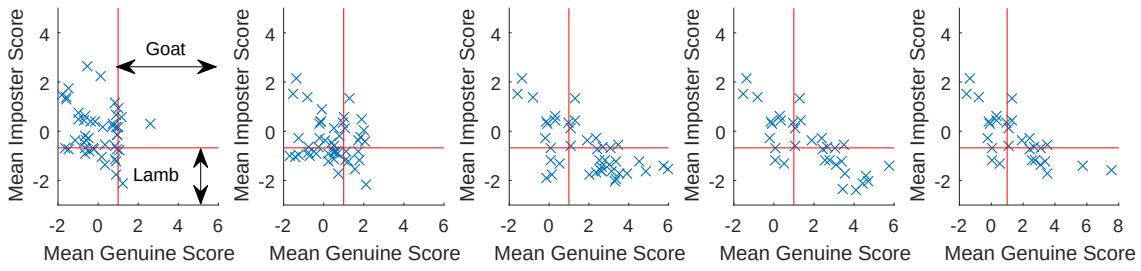
Next, we investigate the impact of the attack on the user groups (or animals) specified by the biometric menagerie. The biometric menagerie is a categorization of users of a biometric system into groups depending on how well or poorly a user performs.

Grouping these animals depends on a set of thresholds that express how well users authenticate on the system. In this work we fix these thresholds that we find to

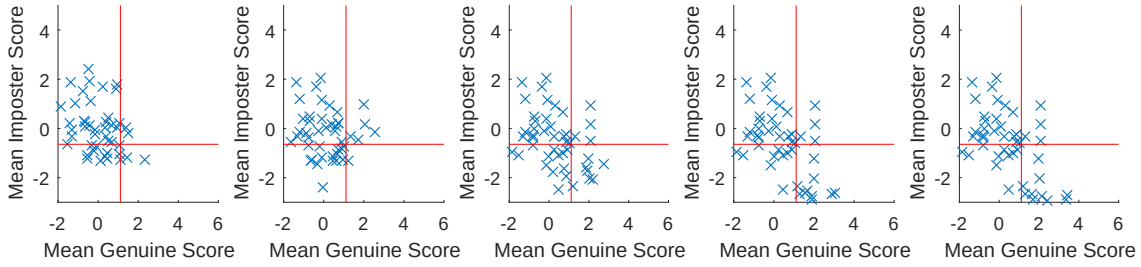
partition the users according to their classification performance. A similar approach was also used in [34]. We classify users whose mean genuine scores are above the 89th percentile as goats, and those whose mean impostor scores are below the 33rd percentile as lambs. Our analysis will focus on these two classes since they generally represent the weak users, whose proportion at different stages of the Frog-Boiling attack will help demonstrate the impact of the attack.

Figure 4.3 captures the attack victims' template transformations as a function of the genuine and impostor scores, after different iterations of the Frog-Boiling attack. The destination template in this case is that of Attacker #1 (User #47). The solid vertical line represents the goats threshold, while the solid horizontal line represents the lambs threshold. For each verifier (i.e., each row), the first graph gives the baseline performance *before* the Frog-Boiling attack was launched, the other plots capturing steps number 50, 100, 200 and 300 of the attack.

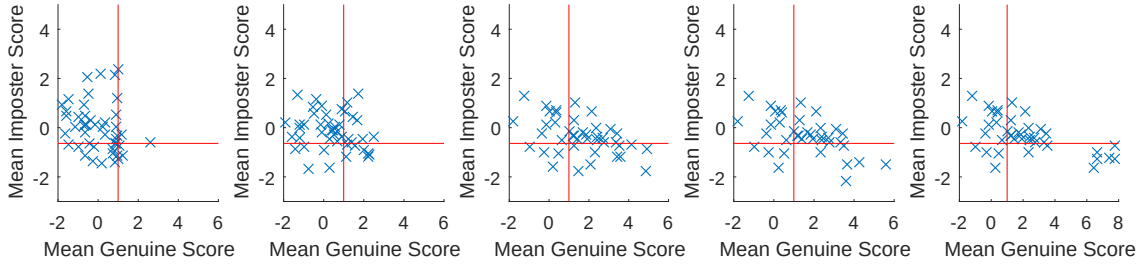
By step number 100, all classifiers see a marked increase in the number of lambs, while an increased number of goats takes a much larger number of steps. This slow increment in the number of goats is likely because the attack employs very small feature modifications, which enables the genuine users to continue to match well against their templates after the first few (say, 100) attacks. On the other hand, the fast increase in the number of lambs could be because a sequence of forged authentication attempts increases the extent of variability of the victim's template, while at the same time decreasing the distance between each pair of genuine and impostor feature vectors. Respective increment and decrement of these two variables then results into a decrease of the impostor scores (see Section 3.2 for score computation formulae).



(a) Scaled Manhattan verifier



(b) Relative verifier



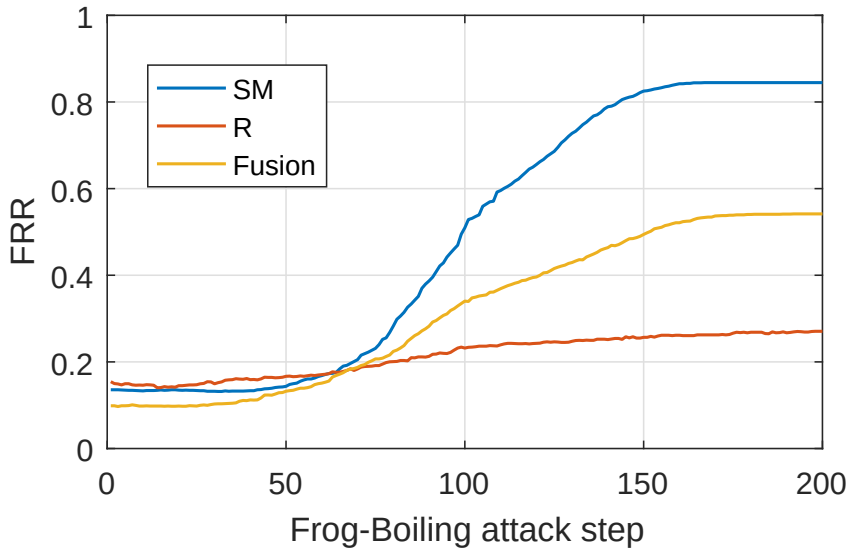
(c) Fusion verifier

Figure 4.3: Menagerie transitions due to the frog-boiling attack launched with Attacker #1's (User #47's) template as the destination template. Each cross represents one of the 46 victims. The right side partition of each graph contains *goats*, while the bottom segment of each graph contains *lambs*. A user could be both a *goat* and a *lamb*. Column 1 to 5 represents the original user status before attack, and the 50th, 100th, 150th, and 200th iterations of Frog-Boiling attacks. All scores shown on the plot are normalized.

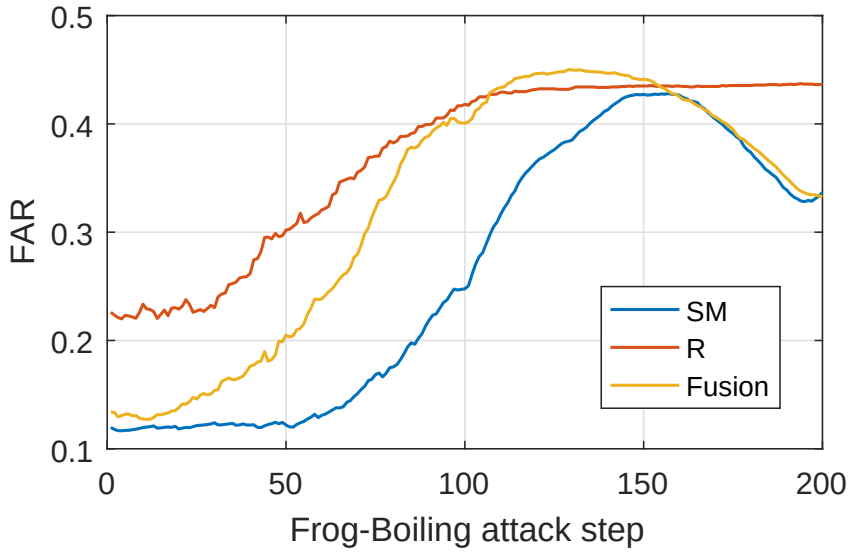
Another attribute of the attack depicted by Figure 4.3 is that some users completely resist the animal transformations (e.g., second user from the top axis of Figure 4.3 (c)), while others see transformations only during the first few stages of the attack (e.g., first user from the top axis). This aspect of the Frog-Boiling attack is crucial for the obfuscation of the attack, since an attack-induced transformation of all victims to a common category/animal (as specified by the *destination* template), would likely ease the task of detecting the attack. In practice however, it is noteworthy that the attack could be tweaked to transform as many users' templates as possible (e.g., by using a greater number of re-authentication attempts during failed attacks, using very small template modifications at each stage, etc.) depending on the aims of the adversary.

A counter-argument that could be raised against the practical implications of the results depicted by Figure 4.3 is that, the 50+ steps required for the attack to cause a significant effect is too high to go undetected in a real system. To address this question, we note that as long as a computer system is compromised by malware, the question of a bot logging into a site multiple times is not an insurmountable challenge today (e.g., see [52]). Additionally, we note that for verifiers built to use 8 to 30 samples during template building (see [26] for a survey), the process of flushing the template should certainly take a much fewer number of steps than those used in this work. We are mostly compelled to use templates containing 100 typing samples (and thus 100+ steps for the attack), because of the large size of the dataset we used.

Up to this point, we have not addressed the question of how long users continue to log into their accounts during the course of the attack. If users fail to log onto



(a) False Rejection Rate (FRR)



(b) False Acceptance Rate (FAR)

Figure 4.4: The changes of average FRR and FAR during the process of the frog-boiling attack, with Attacker #1's profile as the destination template.

Table 4.3: EER of the keystroke verification system under the frog-boiling attack with mixed attempts.

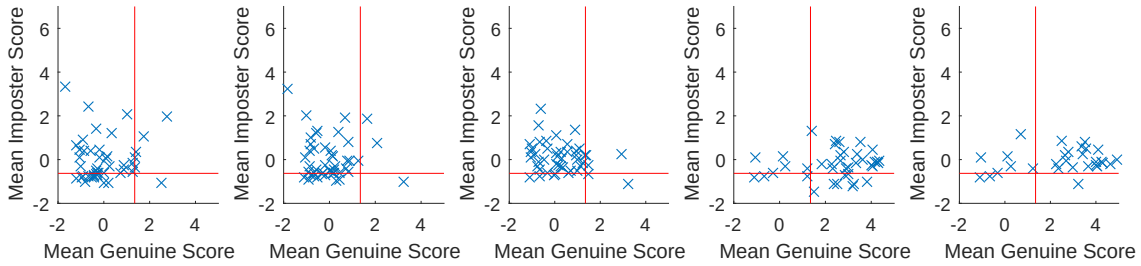
Mixed Rate	Attacker #1			Population Attack		
	SM	R	Fusion	SM	R	Fusion
Mixed 2:1	20.70	19.74	13.91	20.63	19.98	12.11
Mixed 1:1	31.34	24.57	26.08	27.15	22.45	24.55
Mixed 1:2	40.47	28.04	32.53	32.51	26.53	30.84

their accounts after just a few steps of the template transformation process, the attack could be easily detected and precluded in its early stages. On the other hand, if users still access their accounts during the course of the attack, the adversary could expose users' profiles to intrusion by both synthetic and zero-effort impostors (after weakening their templates) for a long time before detection. Figure 4.4 gives some insights into this aspect of the attack, where we show the evolution of system FRR and FAR during the frog-boiling attack. FRR are calculated with the same genuine testing set as in baseline test, and the samples to generate FARs are subtracted from the attacker's own typing sample. For all verifiers, the users represented in Figure 4.4 see very little change in their FRRs up to a number of attacks somewhere between 50 and 100. Meanwhile, the FAR begins to increase sharply after just a few attacks. This trend in error rates shows that the likelihood of an attacker accessing a user's account increases fast while the likelihood that a genuine user fails to access their own accounts only starts to get high after a very large number of attacks. Next, we test the attack performance with the *mixed attack* scenario, and see how the attack performs while victims keep using the system during the attack process.

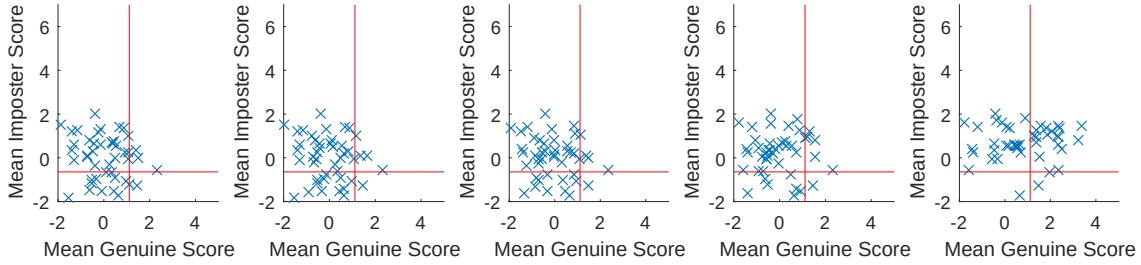
Table 4.3 summarizes the EER performance of the keystroke verification system under the Frog-Boiling attack with both genuine and attack attempts. Although the

EERs are not as high as the results with *pure attack* (Table 4.2), we see an increased EER over the baseline. For example, with Attack #1's template as the destination template, the EER performance of SM verifier has over 200% increment over baseline (from 13.29% to 40.47%), and the Fusion verifier has over 220% increment on EER. Similar to the results in Table 4.2, although the R verifier performs the worst at baseline, it has the lowest EER after the Frog-Boiling attack, due to the nature of the verifier being non-Euclidean. Additionally, we could observe the trend that when attack samples share a larger proportion in all the authentication attempts, the EER after attack is higher, which indicates a greater impact from the frog-boiling attack. This trend shows that the attacker should generate sufficient amount of attack samples in order to create an effective attack. However, a more frequent attack rate would also raise the suspicion, and when the victim's attempts share the majority (in which case the attack would unlikely be detected), the attack has little to no impact to the system.

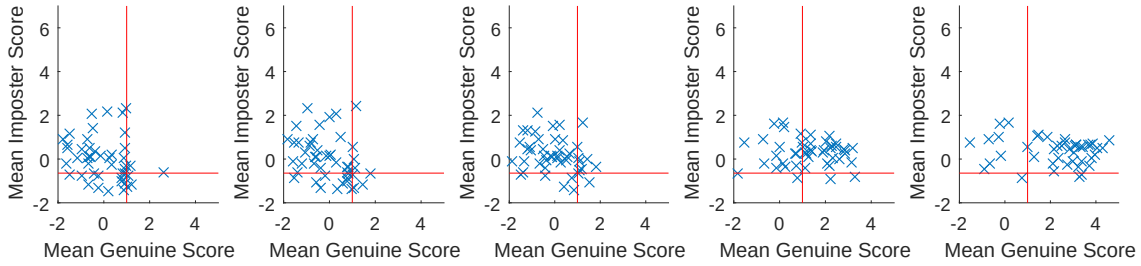
Figures 4.5 and 4.6 show the menagerie transitions due to the frog-boiling attack in *mixed attack* scenario, and the mixed rates are 1:2. Figure 4.5 are generated with Attacker #1's template as the destination template and Figure 4.6 shows the transition with *population attack*. It is observed that the users are moving from the top left section (sheep) to the right section (goat) or bottom section (lamb), a similar trend as shown in Figure 4.3. We see an increased number of goats with all verifiers when the system suffers from 200 time unit of attacks (i.e., the most right column). Comparing with the three verifiers, users have the least menagerie transition with R verifier. This matches the results we have in Table 4.3.



(a) Scaled Manhattan verifier

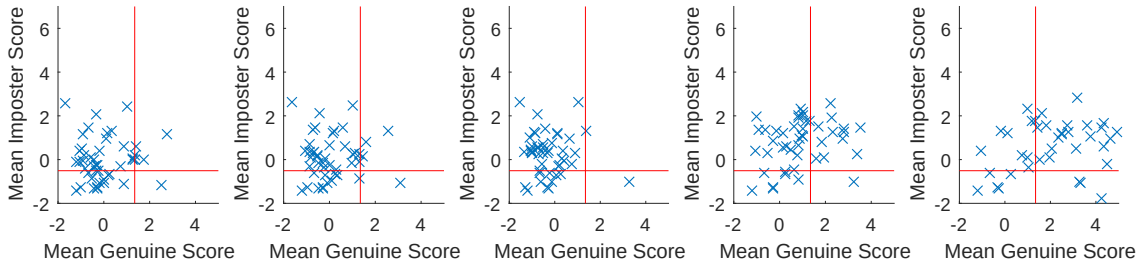


(b) Relative verifier

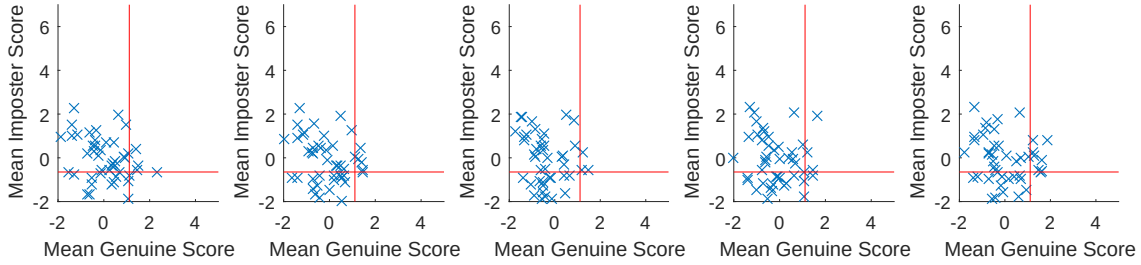


(c) Fusion verifier

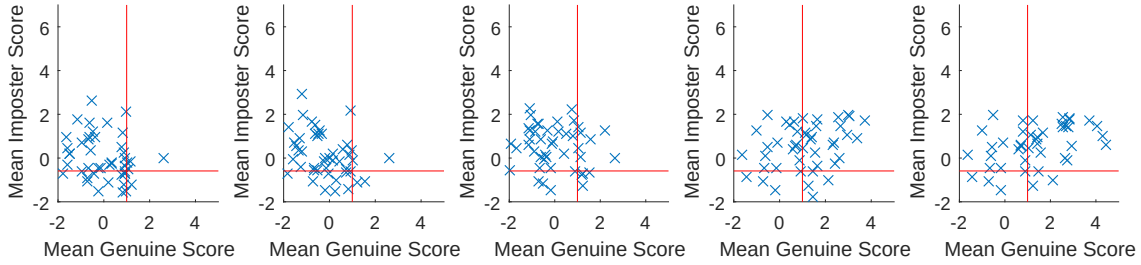
Figure 4.5: Menagerie transitions due to the frog-boiling attack launched with Attacker #1's (User #47's) template as the destination template, with *mixed attack* rate of 1:2.



(a) Scaled Manhattan verifier

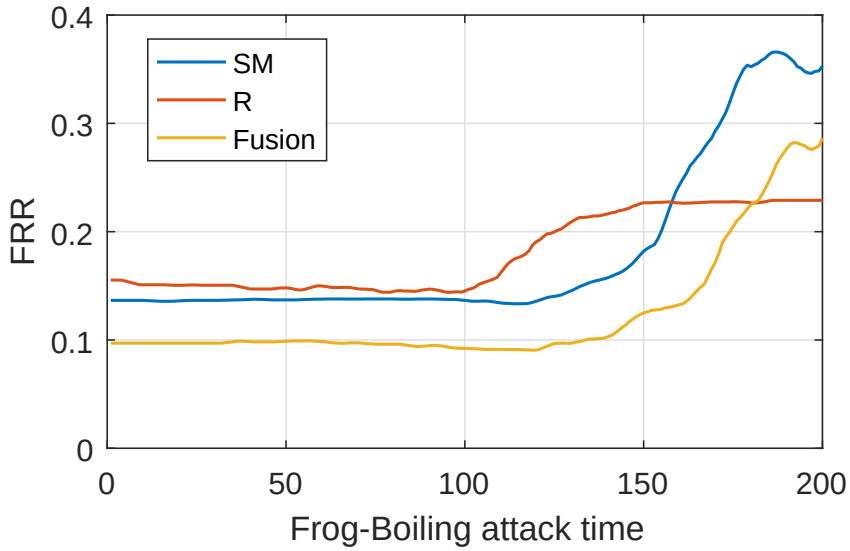


(b) Relative verifier

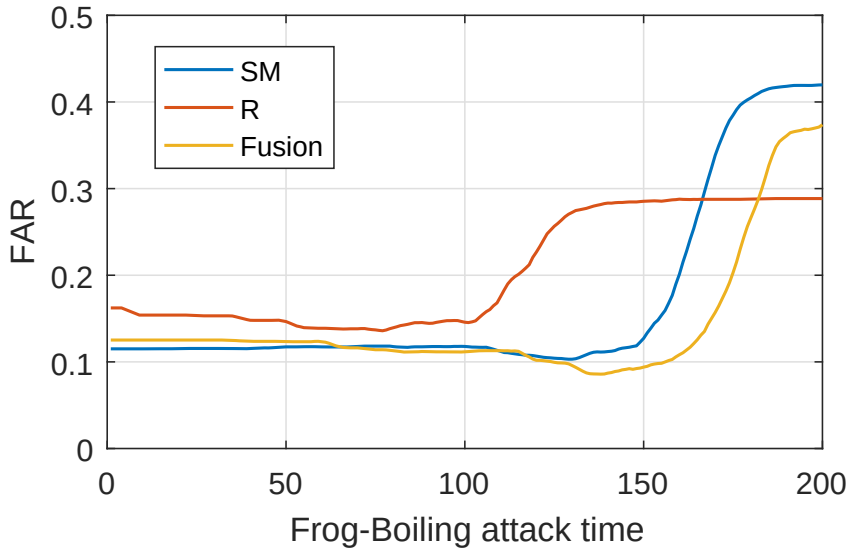


(c) Fusion verifier

Figure 4.6: Menagerie transitions due to the frog-boiling attack launched with the population template as the destination template, with *mixed attack* rate of 1:2.



(a) False Rejection Rate (FRR)



(b) False Acceptance Rate (FAR)

Figure 4.7: The changes of FAR and FRR over time during frog-boiling attack to the keystroke verification system. Each figure shows results of three verifiers with mixed attack, with mixed rate of 1:2.

Figure 4.7 shows the change of FRR and FAR with *mixed attack* (i.e., mix rate of 1:2) from Attack #1. All verifiers see an increased FRR and FAR because of the attack. However, most of the changes occur after 100 time steps. Comparing the results with the *pure attack*, the *mixed attack* takes more attacks to make impact to the system. Although the R verifier (in red) performs the best against the frog-boiling attack, the error rates changed by the attack occur prior to the other two verifiers.

We have shown how the Frog-Boiling attack can effectively intrude a keystroke biometric systems with both error rate increments and menagerie transitions. In the next chapter, we will describe our proposed defensive mechanism, the residual-based detector, against the Frog-Boiling attack.

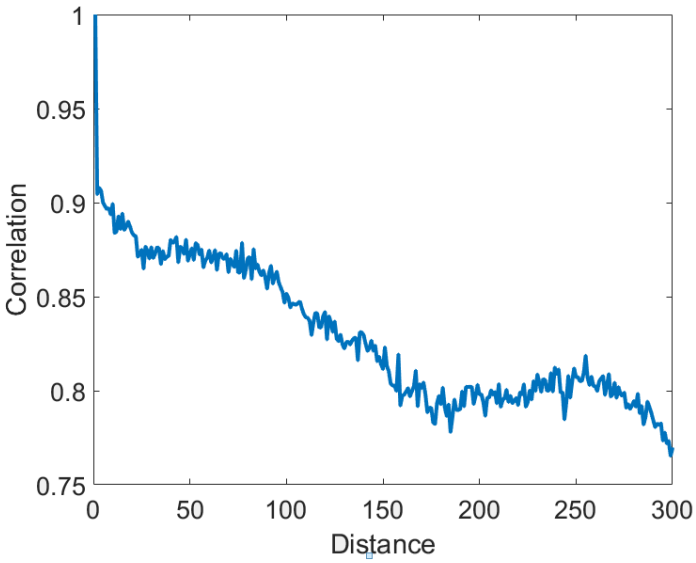
CHAPTER 5

RESIDUAL BASED DETECTOR AGAINST THE FROG-BOILING ATTACK

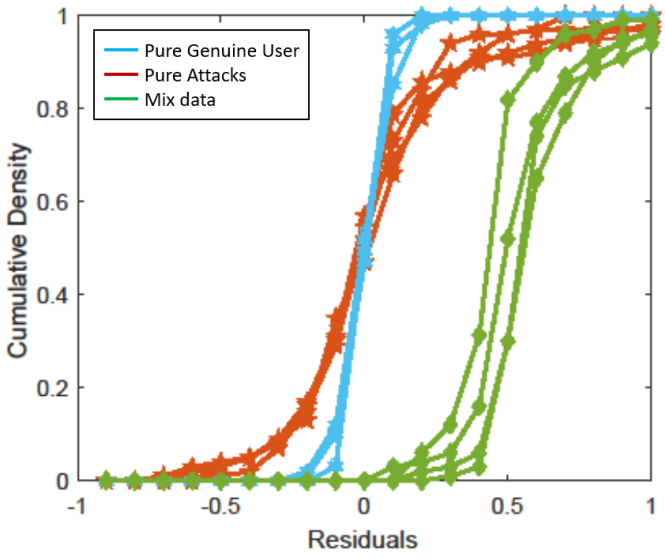
Through the frog-boiling attack, attackers can generate the attacking samples extremely similar to the genuine user templates but with slight differences and thus can gradually drift the original template towards a target template by template updating. So it is imperative to develop an effective defense mechanism which can effectively identify the differences between the genuine user templates and the drifting attacker templates. In this chapter, we will describe a residual-based detector to prevent suspicious updates to the system.

5.1 Residual Distribution Features

Inspired by the existing work in steganography [44], we propose a new approach to detect the frog-boiling attack, specifically to identify those attack samples that have a very high level of similarity as the genuine samples, leveraging the statistical residue distribution features. The rationale of this approach is to capture the intrinsic dependency within the original genuine samples. Such dependency will be altered or even lost when artificial attack samples are synthesized or generated based on the genuine samples. Figure 5.1 (a) depict the average correlation levels corresponding to the varying distances between any two sample points within 300 samples. It is



(a) Correlation between sample points of keystrokes



(b) Cumulative density distribution of residuals for genuine samples, attack samples, and mixed attack samples of keystrokes

Figure 5.1: Correlation and residual distribution for keystroke samples

shown that both types of data show strong correlation (i.e., dependency) among a small set of closely adjacent sample points (e.g., correlation coefficient > 0.9). Such dependency will drop dramatically as the increase of the distance between any two sample points.

The residual among i adjacent sample points is defined as:

$$R_i = \hat{X}_i(N_i) - c\bar{X}_i \tag{5.1}$$

For each feature (n represent the index of the corresponding feature), the first-order and second-order residuals are extracted as follows:

$$R_1^n(i) = X_{i+1}^n - X_i^n \tag{5.2}$$

$$R_2^n(i) = X_{i-1}^n + X_{i+1}^n - 2X_i^n \tag{5.3}$$

X_i^n represents the nth feature value in the ith sample. So the corresponding first order residual $R_1^n(i)$ is calculated by the difference between X_i^n and its following sample's nth feature X_{i+1}^n. The second-order residual $R_2^n(i)$ is the difference between the sum of ith sample's neighbors' nth feature value X_{i-1}^n and X_{i+1}^n minus X_i^n times by 2. Then a probability density distribution is calculated based on k (k is the block size, $k \in N$) residuals in the first and second order. Figure 5.1 (b) shows the cumulative density distribution of the 1st-order residuals for the Feature 18 in the genuine samples of Subject S002 and the frog-boiling attack samples to S002. It is observed that, there are distinguishable differences of the residual distributions among the genuine samples (in blue), the pure attack samples (in red) and the mixed attack samples (in green). It is shown that the residual distribution of the genuine samples has a smoother transition

over a relative larger interval, unlike the sharp transition seen for the attack samples, which indicates the manipulating nature of the attack samples. For each feature of keystroke data, a residual distribution will be generated and used as the inputs of the classifier (to be detailed in the next section).

In our prior work [46], we explored the use of residual distributions to differentiate the genuine ERP samples against the synthesized ERP samples by injecting a very small level of Gaussian noises onto the authentic samples. In this study, residual distribution is adopted as an effective tool to represent the dependency nature of sample points and identify the differences of sample points between the genuine samples and the more demanding frog-boiling attack samples.

5.2 Design of the Residual Based Detector

5.2.1 Random Forest Ensemble Classifier

Due to the high variance in keystroke data and the large dimension of the residual distribution features, a classifier could be easily over-fitted. Moreover, the residual distribution features often possess different importance levels. For example, the residual distributions of some sample features may demonstrate a clear distinguishable difference between the genuine data and the attack data, such type of residual distributions shall play more important role (i.e., have higher weight) in recognizing the attack samples. Therefore, in this study we propose to use the Random Forest Ensemble Classifier (RFEC) which has been proven to be effective in handling high variance, high dimensional data; be resistant to over-fitting; and be able to estimate the feature importance. RFEC is an aggregation of multiple weak classifiers

$\{h(X, \Theta_k), k = 1, 2...K\}$, where Θ_k is the parameter set of the individual decision tree k, and K represents the number of trees. The forming process of a RFEC and principle of feature importance estimation by RFEC are as follows:

1. K subsets are randomly extracted from the original training dataset via bootstrapping [42]. Corresponding to these K subsets, K decision trees are constructed and trained. For each extraction, the subset which is not chosen is named as "Out-of-Bag" (OOB) data (OOB_k);
2. For each decision tree, if there are n features, each time m features are extracted ($m \leq n$). The decision tree keeps choosing and splitting the "most significant" feature ($f \in m$) until it is fully grown;
3. Aggregation at decision-making is realized through the majority voting of these K decision trees with weights w_k ($w_k \propto \frac{1}{E(OOB_k)}$, in which E is the error estimation function);
4. The feature importance estimation is calculated by permuting one feature across OOB data and measuring how worse the MSE of RFEC predictions becomes after the permutation.

Fully growing each decision tree allows RFEC to be capable of processing high dimensional features. The creation of splits is then based on a random set of bootstrap samples, which help reduce variance and avoid over-fitting. The majority voting with w_k aggregation method makes RFEC resistant to over-fitting and effective in handling high variance data. With the importance estimation of the random forest, a filtering and retraining model similar to the gene selection [41] is designed as follows: In every iteration, according to the importance estimation result, part of the less important

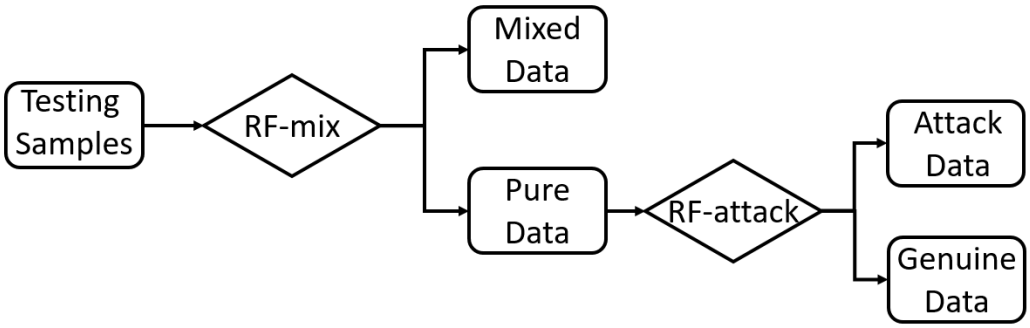


Figure 5.2: The flow diagram of the proposed two-layer defense mechanism

features are filtered out and discarded, a new RFEC is then trained with the remaining feature set. Through the testing of the validation data, the RFEC which performs the best and the corresponding feature set is kept. In this case, for different subjects, the most suitable and important feature set can be identified and chosen for training the classifier.

5.2.2 Two-Layer Defense Mechanism

In some circumstances, data received by the authentication system is mixed with the samples from both the attacker and the genuine user. In this case, a single binary or ternary (three categories including genuine samples, attack samples, and mixed samples) RFEC is not capable of distinguishing the mixed data. However, the transitions from the genuine user to the attacker, and vice versa, can produce some high amplitude residuals which will result in a much wider residual distribution than the one of the genuine user or the attacker. For instance, according to Figures 5.1 (b), it is observed that the mixed data's residual distribution (in green) is remarkably deviated from the distributions of the genuine samples (in blue) and the attack samples (in red). On the other side, although all centered around the value of zero, the residuals

of the genuine samples and the attack samples still demonstrate distinct distributions. Based upon these observations, a two-layer defense mechanism is proposed in this study, as illustrated in Figure 5.2. Any input sample will pass through the first RFEC ("RF-mix") which distinguishes the mixed data and the pure data (including both the genuine and attack samples), and the identified pure data will be further processed by the second RFEC ("RF-attack") to distinguish the genuine data and the attack data.

5.3 Evaluation and Performance of Residual Distribution based Two-Layer Defense Mechanism

To evaluate the effectiveness and efficiency of the proposed two-layer, residual distribution based defense mechanism (see Figure 5.2) for the Frog-Boiling attacks on biometric authentication systems, we develop the evaluation protocols based on the keystroke dynamics biometrics investigated in this study.

5.3.1 Experiment Setting

RF-mix: an RFEC with the goal of distinguishing the mixed data and the pure data.

Training: RF-mix is trained with the mixed data and the pure data for each user. The pure data consists of the genuine keystroke data from sessions 2 to 5 and the attacking samples generated by Gaussian random noises. The mixed samples is created with a mix of both pure data sources. Genuine and attack samples are uniformly random ordered in the training data.

Testing: RF-mix is tested with the genuine samples from sessions 6 to 8, the frog-boiling attack samples and the mixed test data from the general population or a

Table 5.1: Performance of RF-mix, RF-attack and Two-Layer Classifiers

Classifiers	**FRR**	**FAR**							
		Pure Attack		**Mix 1:2**		**Mix 1:1**		**Mix 2:1**	
		pop	**A1**	**pop**	**A1**	**pop**	**A1**	**pop**	**A1**
RF-mix	1.73[a]/1.8[b]	N/A		17.82	16.24	20.34	19.53	23.67	22.58
RF-attack	7.25	1.75	1.37	49.55	50.88	62.39	65.54	70.86	67.45
Two-Layer	8.53	1.37	0.98	14.15	12.80	15.24	16.71	26.83	27.80

Note: [a] for pure genuine and population attack samples, [b] for pure genuine and user-specific attack samples.

specific user (for population attack or user-specific attack, respectively). The mixed test data is generated with a mix of the genuine samples and the attack samples with varying proportions, and is Poisson randomly distributed.

RF-attack: a second RFEC for categorizing data into the genuine or attack samples.

Training: RF-attack is trained with the genuine data and the attack data for each user. With 8 sessions of the genuine keystroke data from each user, we train the RF-attack with sessions 2 to 5 (200 keystroke repetitions for each user) as the genuine training set. Keystroke samples in session 1 are discarded because it is found that the residuals in session 1 are very unstable given the nature of practicing an unfamiliar set of keystrokes. For attack training set, we randomly generate 10 attack training samples from each genuine training sample with Gaussian random noises.

Testing: RF-attack is tested with the genuine samples from session 3 (50 repetitions) and the frog-boiling attack samples. We perform both the user-specific attack and the population attack in the testing. The frog-boiling attack test samples are generated in the same way as Section 4.3.

5.3.2 Experimental Results

Table 5.1 summarizes the performance of the proposed classifiers (including the RF-mix, the RF-attack, and the two-layer structure) when provided with the pure genuine samples, the pure frog-boiling attack samples, and the mixed samples with varying proportions (the ratios between the genuine and the attack samples are $1:1$, $1:2$, and $2:1$, respectively).

The goal of the RF-mix classifier is to distinguish the pure data (including both the genuine and attack samples) and the mixed data, which works as a preprocessing filter to detect and remove the more challenging and demanding mixed data samples. To maximumly retain the genuine samples for updating the authentication system, we give a higher priority to the FRR (i.e., the percentage of pure data including both genuine and attack samples that are falsely labeled as the mixed data and thus discarded) than the FAR (i.e., the percentage of mixed data samples that are falsely labeled as the pure data and are further sent to RF-attack for processing). It is shown that, RF-mix classifier achieves a very low level of FRR ($< 2\%$) as well as an acceptable level of FAR ($\sim 20\%$). It is worthy to note that, given the different goals of RF-mix and RF-attack classifiers, the definitions of FRR and FAR are also different. For RF-attack and the two-layer classifiers, the FRR indicates the amount of the genuine samples (not including attack samples) are falsely recognized as the attack, and in contrast, the FAR represents the amount of the attack samples are falsely accepted by the system.

According to Table 5.1, it is shown that only 1.75% of pure population attack samples are accepted to update users' templates in the keystroke authentication

system. Similar results can be seen for user-specific attacks. When facing the more challenging mixed data including both the genuine and attack samples, unsurprisingly, a significant amount of mixed samples are falsely accepted. The higher the percentage of the genuine samples in the mixed dataset is, the more difficult it is to identify the attack samples. As the mix ratio increases from 1:2 to 1:1 and then to 2:1, the FAR correspondingly increases from 49.55% to 62.39% and then 70.86% for the keystroke system and from 29.87% to 34.67%. Based on this observation, it is clear that the RF-attack classifier itself can effectively distinguish the genuine and attack samples, which however, is less capable when a more challenging mixed dataset is presented. This also proves the necessity of applying another separate filtering mechanism — the RF-mix classifier in our approach.

Combining the aforementioned RF-mix and RF-attack classifiers, the proposed two-layer defense technique shows a very impressive performance in detecting frog-boiling attack samples while retaining the genuine samples. In the keystroke authentication system, FRR is 8.53%, and FAR is 1.37% (population attack) or 0.98% (user-specific attack) when facing the pure attack samples. When the input data is mixed with both genuine and attack samples, the proposed two-layer defense mechanism can still effectively detect and recognize the attack samples most of the time. As explained in Section 4.4, the mixed data with the ratio of genuine samples to attacks which is higher than 1:1 will significantly decrease the frog-boiling effect to the system. That means, if the mixed data is dominated by the genuine samples, it is less likely for the attacker to drift and influence the template updating results of the

target authentication system. Consequently, the FARs at ratio 1:1 and 1:2 are still acceptable for a qualified defensive system.

5.4 Performance Evaluation of Biometric Authentication System Against frog-boiling Attacks

We have demonstrated that our proposed residual distribution based, two-layer defense mechanism can effectively detect and block fake updates to the authentication system, as shown in Section 5.3.2. In this section, we will further evaluate how this mechanism protect and affect the performance of the keystroke authentication systems under frog-boiling attacks.

5.4.1 Experimental Settings

We use a similar experimental setting as in Section 4.3, except that the two authentication systems are equipped with the proposed residual-based defensive mechanism to authenticate the input template updates as well. As described in Section 5.4.2, the systems make an updating decision for every 20 legit incoming samples, and we use the sliding window updating mechanism. We evaluate the two systems with both user-specific attacks and population attacks. For the user-specific attacks, destination templates are generated from Attacker #1 in both biometric systems, the most effective attackers in both cases.

5.4.2 Performance Evaluation on Keystroke Verification System

Table 5.2 summarizes the EER performance of the keystroke verification system with the proposed defense mechanism under the frog-boiling attack. Comparing against the baseline performance (see Table 4.1), the EERs stay almost the same, or even

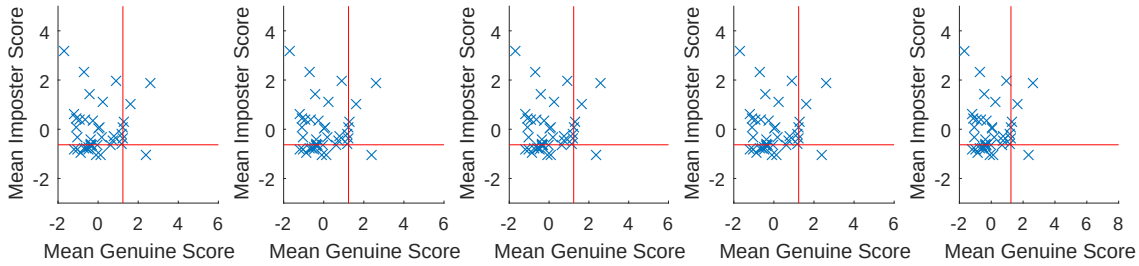
Table 5.2: EER performance (in percentage) of keystroke verification system with the proposed defense mechanism under the frog-boiling attack.

	Attacker #1			Population Attack		
	SM	R	Fusion	SM	R	Fusion
Pure attack	15.33	19.79	10.91	14.21	19.88	9.89
Mixed 1:2	14.65	18.63	9.87	13.32	17.50	9.66
Mixed 1:1	13.78	18.14	9.69	13.11	17.42	9.65
Mixed 2:1	13.57	17.55	9.67	12.96	17.48	9.66

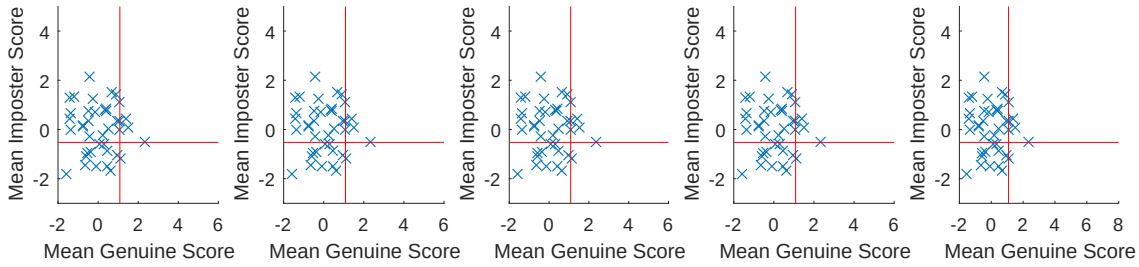
lower than the baseline. For example, the SM verifier has an average EER percentage of 13.29 at baseline, and 12.96%, 13.11%, and 13.32% under the population attack with genuine/imposter attempt proportion of 2:1, 1:1, and 1:2, respectively. The results show that the enhanced keystroke verification system becomes very resistant to the frog-boiling attack and the template updating mechanism can work properly even in the presence of fake samples.

Figure 5.3 shows the menagerie transition when the keystroke system with residual-based detector is under the frog-boiling attack, and the type of attack is *mixed attack* with a rate of 1:2. All the movements of the crosses are insignificant and neglectable, and no animal transition can be observed. This indicates that the frog-boiling attack has limited impacts to transfer good performed users to bad performed users. Similar results are observed for the menagerie transition with *pure attack*, which are thus not presented here due to the space limit.

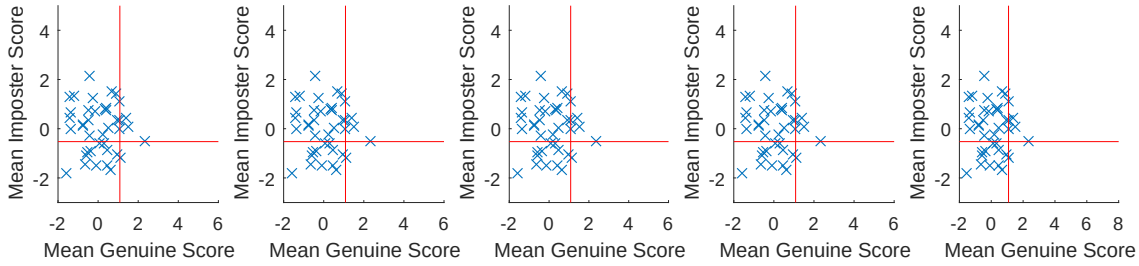
Figure 5.4 shows the FRR and FAR performance of the keystroke verification system with the proposed residual-based defensive mechanism, under the frog-boiling attack (i.e., *mixed attack* with a rate of 1:2). It is observed that the changes to the error rates are very small, and the most remarkable change seen for the SM verifier only increases the FRR from around 0.13 to around 0.16. Some other verifiers



(a) Scaled Manhattan verifier

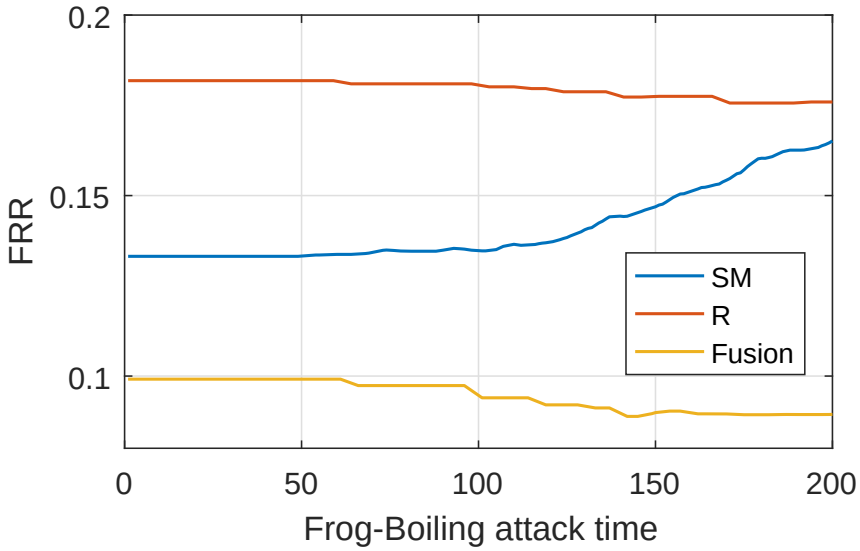


(b) Relative verifier

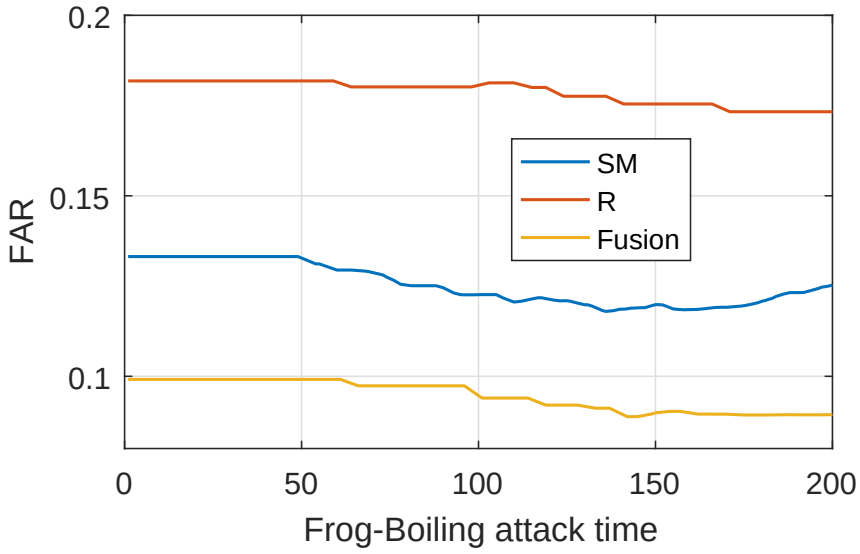


(c) Fusion verifier

Figure 5.3: Menagerie transitions due to the frog-boiling attack launched with Attacker #1's (User #47's) template as the destination template, with *mixed attack* rate of 1:2. The residual-based detector is used in the system.



(a) False Rejection Rate (FRR)



(b) False Acceptance Rate (FAR)

Figure 5.4: The changes of FAR and FRR over time during frog-boiling attack (e.g., *mixed attack* with a rate of 1:2) to the keystroke verification system with the proposed residual-based detector.

even see a decrement in error rates. For example, the FRR and FAR of the fusion verifier decrease from 0.1 to around 0.09. Results in Table 5.2, Figure 5.3 and 5.4 all provide evidences that our residual-based defensive mechanism successfully prevents the intrusion from the frog-boiling attack.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusion

In this dissertation we analyzed the template evolution in keystroke dynamics, investigated an adversarial template drift (i.e. the Frog-boiling attack) and analyzed the trade-off between defending adversarial template drift and ability to update user templates. Our results of template evolution analysis shows that although KIT features have higher difference than KHT features, KHT features seeing more significant change, and the evolution of keystroke templates strongly degrades the EER performance. While our analysis of template evolution provides solid evidence that template updating mechanism is needed, the study of the Frog-boiling attack reveals the risk of adversarial template drift, showing how the attack changes well-performed users to ill-performed users. After adding defensive design in the system (i.e. two-threshold updating) with various scenarios, our analysis gives strong evidence that a system with better effectiveness to update user template has larger vulnerability to adversarial template drift, and shows that the impact of the Frog-boiling attack can be limited while the updating threshold reaches a certain point. We have also introduced a novel detection mechanism that exploits correlation between samples to find spurious update samples.

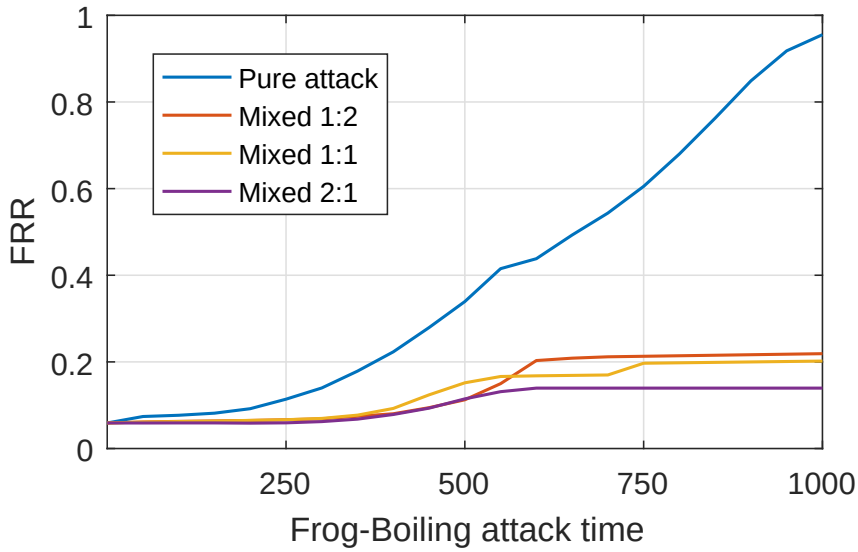
We posit that the attacks and the detection mechanism can be extended to any learning mechanism and thus require serious consideration by the community.

6.2 Future Work

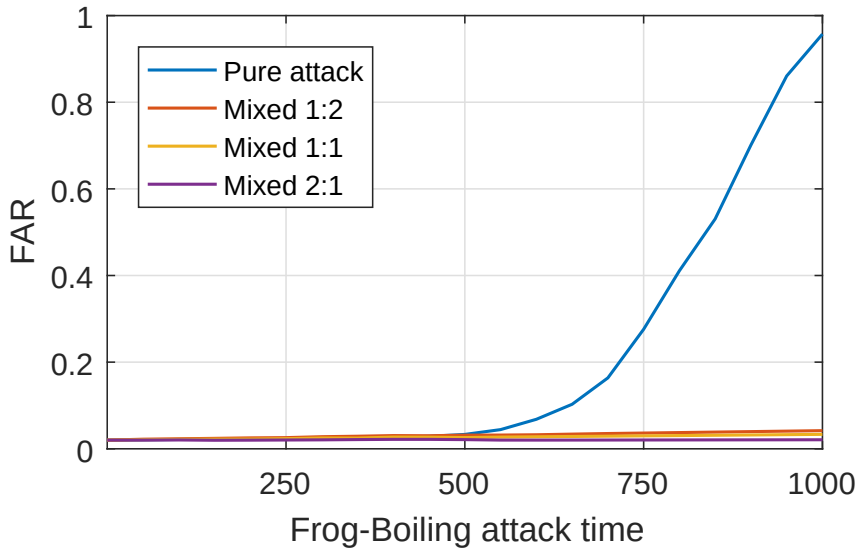
6.2.1 Applications on Other Biometric Modalities

With evidences of biometric aging effect in other biometric modalities in previous studies [27, 28, 29, 15, 47], we believe the applications of our attack and defense mechanism is not limited to keystroke dynamics, and we will extend our work to other modalities in the future. As described in Section 4.1, one of the advantage of our attack mechanism is that it requires minimum knowledge from the attacker. We will keep the same approach when we extend our mechanisms with minor tuning for various biometric systems. As an example of our future work, we sampled an ERP brainwave dataset [53] with some brief results. Some recent studies on brainwave biometric [62] showed the impact of aging effect in EEG data, which indicated that an adaptive system for brainwave authentication is needed.

We tested the Frog-Boiling attack in an ERP brainwave identification system based on cross-correlation [53, 54], with a similar attack design as our keystroke system (see Section 4.1). Figure 6.1 shows the error rate changes of the brainwave identification system due to the influence of the frog-boiling attack. With the pure attack, (the blue lines in both figure), we can see significant increments for both FRR and FAR, from close to 0 to over 90%. With the mixed attack, FRR is increased from around 6% to around 20%, and a slight increment in FAR. An interesting observation from Figure 6.1 is that the increment of FAR with the pure attack starts at around



(a) False Rejection Rate (FRR)



(b) False Acceptance Rate (FAR)

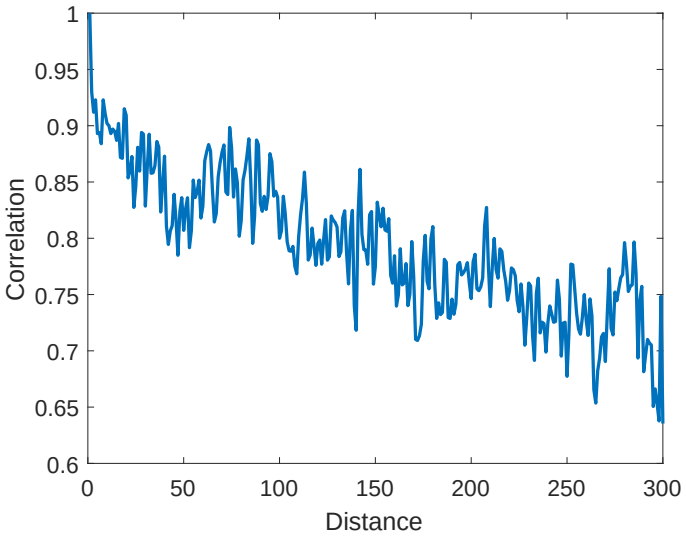
Figure 6.1: The changes of FRR and FAR over time during frog-boiling attack to the brainwave identification system. Attacks are generated with attacker #1.

500th iteration of the frog-boiling attack, this is because after 500 iterations, the attack sample transfers to the attacker's template (the destination template, see Section 4.1 for details about how the attack samples transfer to the attacker's template), and FAR is calculated based on the attacker's brainwave samples. This attack's results indicates the effectiveness of Frog-Boiling attack with ERP brainwave biometric. Next we analyzed the correlation and residual distribution for ERP samples.

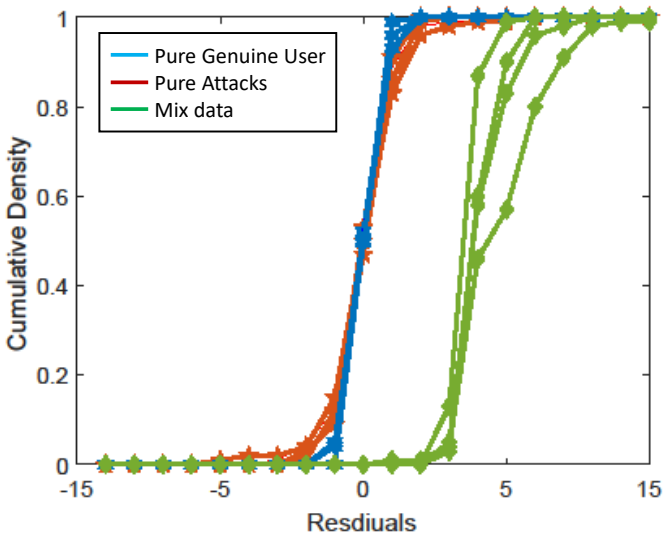
In Figure 6.2, the correlation, the first order and the second-order residual were calculated with the same method described in Section 5.1. Comparing to the results for keystroke in Figure 5.1, similar distinguishable differences among the genuine samples (in blue), the pure attack samples (in red) and the mixed attack samples (in green), can be observed on ERP brainwaves as well. It is shown that the residual distribution of the genuine samples has a smoother transition over a relative larger interval, unlike the sharp transition seen for the attack samples, which indicates the manipulating nature of the attack samples. Figure 6.1 and 6.2 gave us some insights of how our attack and defense mechanisms may work in other biometric modalities and we will perform more throughout tests and analysis in the future.

6.2.2 Potentials of Our Works in Real Life

The goal of studying attacks and defenses in biometrics is to build a more secure, stable, and efficient biometric identification or authentication system. With the growing popularity of various biometric systems [55] in recent years, our works could be extended furthermore in many real-life scenarios. The followings are some examples to apply our works.



(a) Correlation between sample points of ERP brainwaves



(b) Cumulative density distribution of residuals for genuine samples, attack samples, and mixed attack samples of ERP brainwaves

Figure 6.2: Correlation and residual distribution for ERP brainwaves samples

Mobile technologies: Mobile devices (i.e. smartphones, tablets) play a more and more important role in our life, and the security issue raises with the stored personal data. Many modern mobile devices have build-in biometric authentication systems with automatic updates, using fingerprints and face sensor(see [1, 2]), to replace the traditional PIN lock. Some other works built continuous verification with touch sensors and accelerometer on smartphones [56, 57]. These systems are proved to be effective, but they still have space to improve (see a recent reported flaw [58]). Our proposed attack and defense could help enhance these mobile devices by improving their updating procedure and intrusion detection systems.

Car technologies: the technique of self-driving vehicle is a research trend in recent years [59]. One of the greatest challenges for automated cars is the vast variety of vehicle environment and decision making [60], as many noise and outliers in real-world scenarios greatly impact the system. The residual-based mechanism has great potential to filter noise and outliers as many of them are not implications of natural behavior. Our defense procedure could improve the robustness of an automated car.

BIBLIOGRAPHY

[1] Apple. 2017. About Touch ID advanced security technology. https://support.apple.com/en-gb/HT204587. (2017). (accessed July 2019).

[2] Apple. 2017. Face ID Security. https://images.apple.com/business/docs/FaceID_Security_Guide.pdf. (2017). (accessed July 2019).

[3] Ahmed Awad E Ahmed and Issa Traore. A new biometric technology based on mouse dynamics. *IEEE Transactions on dependable and secure computing*, 4(3):165, 2007.

[4] Heikki J Ailisto, Mikko Lindholm, Jani Mantyjarvi, Elena Vildjiounaite, and Satu-Marja Makela. Identifying people from gait pattern with accelerometers. In *Defense and Security*, pages 7–14. International Society for Optics and Photonics, 2005.

[5] Lívia CF Araújo, Luiz HR Sucupira, Miguel Gustavo Lizarraga, Lee Luan Ling, and João Baptista T Yabu-Uti. User authentication through typing biometrics features. *IEEE Transactions on Signal Processing*, 53(2):851–855, 2005.

[6] Francesco Bergadano, Daniele Gunetti, and Claudia Picardi. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):367–397, 2002.

[7] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.

[8] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[9] Eric Chan-Tin, Daniel Feldman, Nicholas Hopper, and Yongdae Kim. The frog-boiling attack: Limitations of anomaly detection for secure network coordinate systems. In *International Conference on Security and Privacy in Communication Systems*, pages 448–458. Springer, 2009.

[10] Rama Chellappa, Charles L Wilson, and Saad Sirohey. Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, 83(5):705–741, 1995.

[11] Sungzoon Cho, Chigeun Han, Dae Hee Han, and Hyung-Il Kim. Web-based keystroke dynamics identity verification using neural network. *Journal of organizational computing and electronic commerce*, 10(4):295–307, 2000.

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

[13] John Daugman. High confidence recognition of persons by rapid video analysis of iris texture. In *Security and Detection, 1995. European Convention on*, pages 244–251. IET, 1995.

[14] George Doddington, Walter Liggett, Alvin Martin, Mark Przybocki, and Douglas Reynolds. Sheep, goats, lambs and wolves: A statistical analysis of speaker performance in the nist 1998 speaker recognition evaluation. Technical report, DTIC Document, 1998.

[15] Samuel P Fenker and Kevin W Bowyer. Analysis of template aging in iris biometrics. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 45–51. IEEE, 2012.

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

[17] The tale of a boiling frog. http://awesci.com/the-old-tale-of-a-boiling-frog. (2014). (accessed July 2019).

[18] Romain Giot, Bernadette Dorizzi, and Christophe Rosenberger. Analysis of template update strategies for keystroke dynamics. In *2011 IEEE Workshop on Computational Intelligence in Biometrics and Identity Management (CIBIM)*, pages 21–28. IEEE, 2011.

[19] Romain Giot, Christophe Rosenberger, and Bernadette Dorizzi. A new protocol to evaluate the resistance of template update systems against zero-effort attacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 131–137, 2013.

[20] Michela Goffredo, Imed Bouchrika, John N Carter, and Mark S Nixon. Self-calibrating view-invariant gait biometrics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(4):997–1008, 2010.

[21] Sajjad Haider, Ahmed Abbas, and Abbas K Zaidi. A multi-technique approach for user identification through keystroke dynamics. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 2, pages 1336–1341. IEEE, 2000.

[22] Anil Jain, Lin Hong, and Ruud Bolle. On-line fingerprint verification. *IEEE transactions on pattern analysis and machine intelligence*, 19(4):302–314, 1997.

[23] Xudong Jiang and Wee Ser. Online fingerprint template improvement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1121–1126, 2002.

[24] Pilsung Kang, Seong-seob Hwang, and Sungzoon Cho. Continual retraining of keystroke dynamics based authenticator. In *International Conference on Biometrics*, pages 1203–1211. Springer, 2007.

[25] R Khandaker, K Balagani, and V Phoha. Making impostor pass rates meaningless: A case of snoop-forge-replay attack on continuous cyber-behavioural verification with keystrokes. In *Proceedings of the IEEE Computer Society and IEEE Biometrics Council Workshop on Biometrics (BIOM)*, 2011.

[26] Kevin S Killourhy and Roy A Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 125–134. IEEE, 2009.

[27] Andreas Lanitis, Chrisina Draganova, and Chris Christodoulou. Comparing different classifiers for automatic age estimation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):621–628, 2004.

[28] Andreas Lanitis. A survey of the effects of aging on biometric identity verification. *International Journal of Biometrics*, 2(1):34–52, 2009.

[29] Andreas Lanitis, Christopher J. Taylor, and Timothy F Cootes. Toward automatic simulation of aging effects on face images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):442–455, 2002.

[30] Anthony J Mansfield and James L Wayman. *Best practices in testing and reporting performance of biometric devices*. Centre for Mathematics and Scientific Computing, National Physical Laboratory Teddington, Middlesex, UK, 2002.

[31] Fabian Monrose, Michael K Reiter, and Susanne Wetzel. Password hardening based on keystroke dynamics. *International Journal of Information Security*, 1(2):69–83, 2002.

[32] Javier Ortega-Garcia, Julian Fierrez, Fernando Alonso-Fernandez, Javier Galbally, Manuel R Freire, Joaquin Gonzalez-Rodriguez, Carmen Garcia-Mateo, Jose-Luis Alba-Castro, Elisardo Gonzalez-Agulla, Enrique Otero-Muras, et al. The multiscenario multienvironment biosecure multimodal database (bmdb). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1097–1111, 2010.

[33] Arun Ross and Anil Jain. Information fusion in biometrics. *Pattern recognition letters*, 24(13):2115–2125, 2003.

[34] Arun Ross, Ajita Rattani, and Massimo Tistarelli. Exploiting the doddington zoo effect in biometric fusion. In *Biometrics: Theory, Applications, and Systems, 2009. BTAS'09. IEEE 3rd International Conference on*, pages 1–7. IEEE, 2009.

[35] Abdul Serwadda, Vir V Phoha, and Ankunda Kiremire. Using global knowledge of users' typing traits to attack keystroke biometrics templates. In *Proceedings of the thirteenth ACM multimedia workshop on Multimedia and security*, pages 51–60. ACM, 2011.

[36] Michael Trusov, Anand V Bodapati, and Randolph E Bucklin. Determining influential users in internet social networks. *Journal of Marketing Research*, 47(4):643–658, 2010.

[37] Umut Uludag, Sharath Pankanti, Salil Prabhakar, and Anil K Jain. Biometric cryptosystems: issues and challenges. *Proceedings of the IEEE*, 92(6):948–960, 2004.

[38] Zibo Wang, Abdul Serwadda, Kiran S Balagani, and Vir V Phoha. Transforming animals in a cyber-behavioral biometric menagerie with frog-boiling attacks. In *Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on*, pages 289–296. IEEE, 2012.

[39] Juefei Xu, Miriam Cha, Joseph L Heyman, Shreyas Venugopalan, Ramzi Abiantun, and Marios Savvides. Robust local binary pattern feature sets for periocular biometric identification. In *Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on*, pages 1–8. IEEE, 2010.

[40] Neil Yager and Ted Dunstone. The biometric menagerie. *IEEE transactions on pattern analysis and machine intelligence*, 32(2):220–230, 2010.

[41] Ramón Díaz-Uriarte and Sara Alvarez De Andres. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3, 2006.

[42] Bradley Efron and R.J. Tibshirani. *An Introduction to the Bootstrap (Monographs on Statistics & Applied Probability)*. Chapman & Hall, Inc., 1993.

[43] Benjamin I. P. Rubinstein, Blaine Nelson, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J. D. Tygar. ANTIDOTE: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 1–14. ACM, 2009.

[44] Jessica Fridrich and Jan Kodovsky. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3):868–882, 2012.

[45] Marta Gomez-Barrero, Javier Galbally, Julian Fierrez, and Javier Ortega-Garcia. Face verification put to test: A hill-climbing attack based on the uphill-simplex algorithm. In *Proceedings of the 5th IAPR/IEEE International Conference on Biometrics (ICB)*, pages 40–45, 2012.

[46] Qiong Gui, Wei Yang, Zhanpeng Jin, Maria V Ruiz-Blondet, and Sarah Laszlo. A residual feature-based replay attack detection approach for brainprint biometric systems. In *Proceedings of the IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2016.

[47] John Harvey, John Campbell, and Andy Adler. Characterization of biometric template aging in a multiyear, multivendor longitudinal fingerprint matching study. *IEEE Transactions on Instrumentation and Measurement*, 68(4):1071–1079, 2018.

[48] Giulio Lovisotto, Simon Eberz, and Ivan Martinovic. Biometric backdoors: A poisoning attack against unsupervised template updating. *arXiv preprint arXiv:1905.09162*, 2019.

[49] Emanuele Maiorana and Patrizio Campisi. Fuzzy commitment for function based signature template protection. *IEEE Signal Processing Letters*, 17(3):249–252, 2010.

[50] Emanuele Maiorana, Gabriel E Hine, and Patrizio Campisi. Hill-climbing attack: Parametric optimization and possible countermeasures. An application to on-line signature recognition. In *2013 International Conference on Biometrics (ICB)*, pages 1–6. IEEE, 2013.

[51] Enrique Argones Rua, Emanuele Maiorana, Jose Luis Alba Castro, and Patrizio Campisi. Biometric template protection using universal background models: An application to online signature. *IEEE Transactions on Information Forensics and Security*, 7(1):269–282, 2012.

[52] Yao Zhao, Yinglian Xie, Fang Yu, Qifa Ke, Yuan Yu, Yan Chen, and Eliot Gillum. BotGraph: Large scale spamming botnet detection. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, volume 9, pages 321–334, 2009.

[53] Maria V Ruiz-Blondet, Zhanpeng Jin, and Sarah Laszlo. CEREBRE: A novel method for very high accuracy event-related potential biometric identification. *IEEE Transactions on Information Forensics and Security*, 11(7):1618–1629, 2016.

[54] Blair C Armstrong, Maria V Ruiz-Blondet, Negin Khalifian, Kenneth J Kurtz, Zhanpeng Jin, and Sarah Laszlo. Brainprint: Assessing the uniqueness, collectability, and permanence of a novel method for ERP biometrics. *Neurocomputing*, 166:59–67, 2015.

[55] Angelo Genovese, Enrique Muñoz, Vincenzo Piuri, and Fabio Scotti. Advanced biometric technologies: Emerging scenarios and research trends. In *From Database to Cyber Security*, pages 324–352. Springer, 2018.

[56] Abdul Serwadda, Vir V Phoha, and Zibo Wang. Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms. In *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–8. IEEE, 2013.

[57] Abdul Serwadda, Vir V Phoha, Zibo Wang, Rajesh Kumar, and Diksha Shukla. Toward robotic robbery on the touch screen. *ACM Transactions on Information and System Security (TISSEC)*, 18(4):14, 2016.

[58] BBC. 2019. Samsung: Anyone's thumbprint can unlock Galaxy S10 phone. https://www.bbc.com/news/technology-50080586. (2019). (accessed December 2019).

[59] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[60] Supriya B Sarkar and B Chandra Mohan. Review on autonomous vehicle challenges. In *First International Conference on Artificial Intelligence and Cognitive Computing*, pages 593–603. Springer, 2019.

[61] Battista Biggio. Machine learning under attack: Vulnerability exploitation and security measures. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pages 1–2, 2016.

[62] Emanuele Maiorana and Patrizio Campisi. Longitudinal evaluation of eeg-based biometric recognition. *IEEE Transactions on Information Forensics and Security*, 13(5):1123–1138, 2017.

[63] Paulo Henrique Pisani, Abir Mhenni, Romain Giot, Estelle Cherrier, Norman Poh, André Carlos Ponce de Leon Ferreira de Carvalho, Christophe Rosenberger, and Najoua Essoukri Ben Amara. Adaptive biometric systems: Review and perspectives. *ACM Computing Surveys (CSUR)*, 52(5):1–38, 2019.

[64] Gorazd Praprotnik and Nikola Pavešić. The impact of template aging on the performance of automatic fingerprint recognition. *Revija za kriminalistiko in kriminologijo/Ljubljana*, 67(4):371–388, 2016.

[65] Indrė Žliobaitė, Albert Bifet, Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Machine Learning*, 98(3):455–482, 2015.

[66] Abir Mhenni, Estelle Cherrier, Christophe Rosenberger, and Najoua Essoukri Ben Amara. Adaptive biometric strategy using doddington zoo classification of users keystroke dynamics. In *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 488–493. IEEE, 2018.