Implementing Keystroke Dynamics

A Flexible Solution Using K-Means Clustering and ASP.NET MVC

James R. Santiago
The Pennsylvania State University
College of Information Sciences and Technology
305 Information Sciences and Technology Building
University Park, PA 16802-6823
james.r.santiago@gmail.com

ABSTRACT

This paper discusses a project to implement a Keystroke Dynamics system to enhance authentication security. A practical solution is developed over a standard ASP.NET MVC template. Specifically Keystroke Dynamics has been applied to the username and password authentication method by measuring keystroke timing during the input of a users password. Timing is done through a Silverlight control to gain accuracy of key presses to within a millisecond using a high resolution timer.

The distance between keystroke data is calculated using an Euclidean distance. The user's keystroke behavior is stored as a centroid of all successfully authenticated keystroke data. Subsequent logins compare keystroke data with the user's stored centroid using a distance threshold. The actions to be taken upon failure to match the threshold can be decided by either user or site administrator. Flexibility lies in the systems ability to be configured to match the user environment and without needing special equipment or having to have timing software loaded on a users workstation.

The project currently is in a development phase and therefore does not have a sufficient amount of data to support conclusions on false acceptance or false rejection rates. The prime issue is the practical solutions to implementation concerns.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; I.5.3 [**Computing Methodologies**]: Pattern Recognition—*Clustering*

General Terms

Algorithms, Security

Keywords

Keystroke Dynamics, Authentication, K-Means, Clustering, ASP.NET, MVC

1. INTRODUCTION

With increasingly sophisticated attacks on authentication the username and password methodology is becoming less and less accepted as sufficient in protecting access to an information system. Many systems now incorporate a multi-factor authentication method. This comes from the idea that we have three ways to authenticate a user: using something we know, something we have or something we are. The multi-factor method comes in when two or more of these authentication types are combined. Our goal with Keystroke Dynamics is to combine something we know (a password) with something we are (our keystroke behavior).

Keystroke Dynamics is the measurement of the way an individual types on a keyboard. There are several measurements that can be taken which are the force of each key press, the elapsed time that a key is held down and the elapsed time to press the next key. With this behavior based data we can build a profile after several samples are provided by a user. This profile would be defined by clustering algorithm and future samples could be identified as belonging to a particular user by their assignment to a particular cluster.

This is where Keystroke Dynamics comes in with authenticating a user. By checking new data samples against the centroid of the users Keystroke data we can determine if the new data sample is close enough to the centroid to be classified as being similar to a user's typing behavior.

2. IMPLEMENTATION APPROACH

There are several key hurdles in designing an authentication method that utilizes Keyboard Dynamics. These hurdles stem from the computing environment in mind when designing the system. Are you able to supply special keyboards to users? Can you install software directly on the users workstation? Can you restrict the operating system or Internet browser used? In many environments the answer to these questions is no, especially if the computing environment is public or one that has a large user base.

The desired solution should be independent of the hardware or software used by the user. Unfortunately, this requires that we drop the use of pressure sensitive data as it would

require a special keyboard. This reduces the sensitivity of the biometrics data considerably but requiring a special keyboard is not practical for the majority of computing environments. However, the upside to a flexible solution is the the inclusion of a wider range of systems and little to no effort on the user end to adapt from a username and password system to one that also utilizes Keystroke Dynamics.

2.1 Architecture Overview

As shown in figure 1 the Keystroke Dynamics module is added onto the existing username and password architecture. The original authentication steps as well as ASP.NET's membership tables are left unchanged. The major change to the authentication process is that before a user is authenticated they are sent through the Keystroke Dynamics portion. The Keystroke Dynamics data is passed to the account controller along with the username and password. After the username and password are validated the Euclidean distance is calculated between the Keystroke Dynamics data provided and the stored centroid that belongs to the username. If the distance is within an acceptable limit the user will be authenticated and then the keystroke data will be assigned to the user's cluster and the K-means update step will be performed to calculate a new centroid for the user. If the distance is not acceptable then the account controller will take one of two steps. In passive mode the user will still be authenticated but a message will be sent to the user and administrator as a notification of a possible intrusion. Alternately, in active mode the user will not be authenticated.

2.2 K-Means Steps

The K-Means algorithm is not used entirely in the authentication steps but only later during analysis of all logins. Initially the assignment step is simplified so that subsequent logins are added into a users cluster one at a time. The update step follows each login and recalculates the new centroid. Since the login process does not require a repeat of the assignment and update steps and with only a single cluster being calculated at one time the computational complexity becomes simpler than the actual K-Means algorithm such that the running time becomes $0(n)$ linear time.

Additionally, the update step to calculate the new centroid can be ran on a separate thread or queued into a thread pool so as to not slow down the authentication process. Tests so far have not shown the update step to add a significant amount of computation time but since it is not a requirement of authentication the update step can be separated to improve performance.

As stated, the K-Means algorithm can be used in its entirety to analyze logins either offline or from an administrator section of the website. A benefit of performing this analysis can be to identify keystroke data that clashes with other users. This might indicate the existence of a duplicate user or help to determine reasons for false rejection or acceptance rates. I have yet to modify the project to allow for storage of failed keystroke data but the data would be vital to determining the best euclidean distance thresholds that would maximize the true rejection and acceptance rates. Currently only a rough estimate was used to determine at what distance a user should be accepted or rejected.

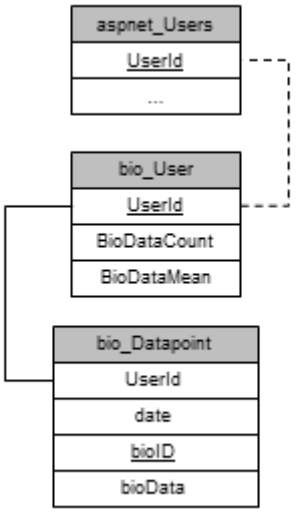

Figure 2: The tables and relations describing the data storage for keystroke data.

3. USING ASP.NET MVC

To demonstrate how Keystroke Dynamics can be easily integrated into authentication I have chosen a web based account system. One of the newer web technologies that has been gaining ground is the Model View Controller (MVC) architecture. To put it very simply, MVC is a programming structure that compartmentalizes the major functions of web providers. The MVC structure can be developed from many web based programming languages such as Java, PHP or in this case ASP.NET.

The main reason why I have chosen to use an MVC architecture is its ability to easily define database structures and perform quick actions on data without overloading the views presented to the user. A built in starter site in the development environment Visual Studio was used as the foundation. This includes a basic username and password authentication system which utilizes ASP.NETs Membership assembly. The Keystroke Dynamics module is then added onto this site. In fact the biometrics aspect of the site can be added on to a fully developed site at any time without need for recreating user accounts.

3.1 Model Design

In the MVC framework the model is the programmatic definition of a data structure. In this project Microsoft SQL was used but various database technologies can be used without having to modify the controllers or views as long as the data structure is queryable with .NETs Linq framework. This project is still a work in progress so the data structure is missing definitions for failed keystroke data. The missing definitions notwithstanding, the model will be fairly simple.

In addition to the model definition a class named BioDataRepository was created to perform functions on the model such as retrieving all bio_User records or adding new bio_Datapoint records. An interface named IBioDataRepository was

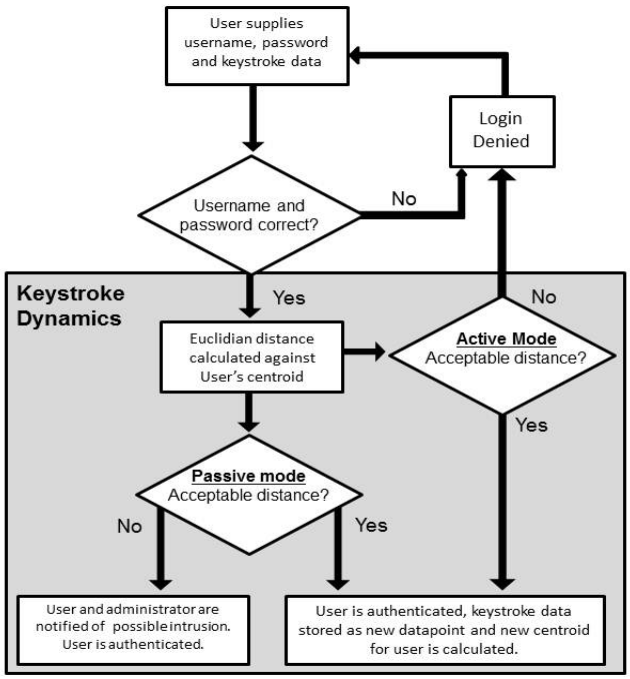

Figure 1: The normal authentication process with the addition of Keystroke Dynamics.

also created that the BioDataRepository implements so that actions on the Keystroke Dynamics model are explicitly defined. The data repository class does an excellent job in simplifying functions, making SQL queries transparent. The model in this project is a Linq to SQL model. The following actions can be performed on the bio_User and bio_Datapoint tables:

```
IQueryable<bio_Datapoint> GetAllDatapoints();
IQueryable<bio_User> GetAllBioUsers();
String[] GetAllBioDatapointsForUser_
    (object ProviderUserKey);
void AddBioUser(bio_User user);
bio_User GetBioUser(object ProviderUserKey);
string GetBioUserDataMean_
    (object ProviderUserKey);
void ResetBioUser(object ProviderUserKey);
int GetBioUserDataCount_
    (object ProviderUserKey);
void DeleteBioUser(object ProviderUserKey);
void AddBioDatapoint(bio_Datapoint dp);
void Save();
```

3.2 Controller Design

The controller handles HTTP actions sent from the browser according to the ASP.NET route table. The original routes defined by the ASP.NET MVC template is kept the same.

The route table defines all HTTP actions as {controller}/{action}/{id}. For example an HTTP_GET request would produce the following code given the request of:

http://domain.com/account/LogOn

```
public ActionResult LogOn()
{
    return View();
}
```

This will produce a HTML page defined by the LogOn view. Separate ActionResults are defined for different HTML actions. In another example the HTTP_POST sent to the LogOn action would attempt to authenticate the user with the form data passed that contains the username, password and keystroke data.

3.2.1 The Account Controller

The Account controller handles user authentication and registration. The HTTP_POST request sent to the LogOn action has been modified so that Keystroke Dynamics is added into the authentication process as shown in figure 1. During registration a call to create a bio_User record has been added in addition to the ASP.NET MembershipUser registration. The validation errors returned to the LogOn view after Keystroke Dynamics fails to authenticate a user are currently informative. This means that the user will be notified that it was Keystroke Dynamics that was not successful if the username and password is correct but the keystroke data is not. This is for testing purposes. It would be advised to restrict the amount of information presented to the user related to authentication failures.

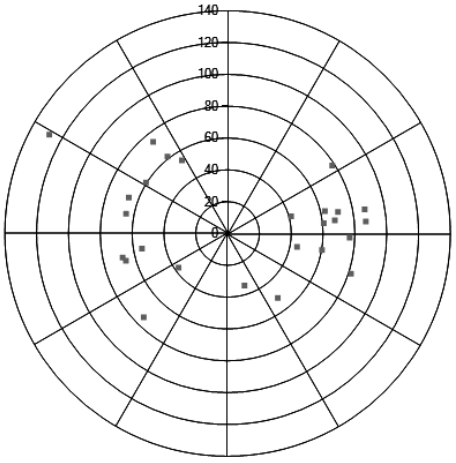

Figure 3: An example of a user's biometrics profile chart. Each datapoint represents a single login and its distance from the user's centroid.

3.2.2 The Home Controller

The Home controller as defined by the MVC template handles basic views such as the index and about page. This controller has been modified to additionally present the user with a view detailing their biometrics profile. The biometrics profile view also allows the user to change their biometrics mode to active or passive. Additionally, an action was created to return an image of a chart detailing the distance of each recorded login from the users centroid. The chart is a polar coordinate plot where the angle of each data point is a random integer and the distance is the Euclidean distance from the user's centroid. The purpose of this chart is to give the user an idea of the accuracy of their keystroke data so that they can make an informed decision regarding the Biometrics Mode that suits their keystroke behavior. The home controller also has an action that handles a request from the user to reset their biometrics data. This was included in the case that the user has significantly changed their keystroke behavior which may have changed after an improvement in typing skill or after a physical change such as arthritis or the loss of a finger.

3.2.3 The Admin Controller

The Admin controller was not included with the MVC template but was added to allow for a designated administrator to review and modify the security functionality of the site without having to modify hard coded features. At the moment only the review of user biometric profiles has been added into the Admin controller. However, the Admin controller will be vital for the future inclusion of Keystroke Dynamics analysis and configuration changes to authentication policies.

3.3 View Design

The view defines the html pages that are presented to the user. Model, view and session data can be passed to the view from the controller to be presented within html elements. In-line code is also available to be compiled at each request for the view. For example in the site.master view a block of code determines whether to present the biometrics profile link to the user according to login status. For this project views for admin pages and for the biometrics profile have been created. CSS formatting and general html styling has been kept the same as the MVC template.

3.4 Localizing a Login Control with Silverlight

One of the major hurdles in providing a flexible Keystroke Dynamics solution is the method of measuring keystroke data without having to install software on the users system. The solution to this hurdle is to localize the code needed to perform the measurements. To localize means to run code directly on the clients machine instead of the server. There are several technologies that can be deployed to meet this requirement such as JavaScript, a Java Applet, Adobe Flash, a localized ASP.NET control or Silverlight. All of these can be defined as a sandbox which is an environment that allows isolated client side resources to be used but with restricted access so that the code running in the sandbox cannot compromise other running applications or client resources such as the file system.

JavaScript was first used but timing was found to be unable to handle one millisecond counting. Different browsers also compile and run JavaScript code differently so one might be counting in units of two milliseconds and another in ten milliseconds. This can cause a false authentication action if the user changes browsers. Also, using larger units of measurement has the effect of lowering the sensitivity of the data which lowers the systems ability to identify a user by their keystroke data.

Silverlight was used next to fix the issues with JavaScript. With Silverlight I had access to a high resolution timer to accurately count in units of one millisecond regardless of the browser used. A high resolution timer is usually defined as timing through direct access to a processors clock. During testing however, I was unable to get a true high resolution response. I used the Stopwatch class found within the Silverlight System.Diagonostics.Threading namespace. The Stopwatch class is fairly new and was programmed with the Windows Phone in mind which may explain the reason why the timer defaulted to system timing instead of using high resolution timing. The Stopwatch class can still be used in a web based Silverlight control by referencing the Microsoft.Phone assembly that is located in the Windows Phone SDK.

Keystroke measurements are taken at key press events. Two measurements are take for each key, one for when the key is pressed and another when the key is released. The first time can be classified as the time taken for a user to move from one key to the other and the second time would be the dwell time on each key. These measurements are passed to the account controller as a string of space delimited integers for easy storage within a SQL database.

4. INTRUSION DETECTION

The prime use of Keystroke Dynamics at this point of development is actually intrusion detection and not authentica-

tion. Through many studies and various implementations of Keystroke Dynamics the results were not accurate enough to be considered worthy of being an authentication process by itself[1]. Therefore it would not be accurate to portray the use of Keystroke Dynamics along with an accurate authentication process as being a two-factor authentication. However, Keystroke Dynamics does shine in other areas such as improving the effectiveness of an authentication that relies on something that can be stolen such as a password or card. In such a scenario Keystroke Dynamics can alert of or prevent a malicious authentication attempt.

In keeping with a flexible approach this project has allowed the user to change the behavior of Keystroke Dynamics during authentication.

4.1 Passive

Restricting authentication on the basis of a user's keystroke behavior might not be applicable in every situation. A user may have an inconsistent typing behavior, share a login with another user (where policy permits) or where the security of a login is not as important as the need for high availability. In such cases a method to simply notify the user and/or administer when a possible intrusion occurs would be the ideal solution. Passive mode is the default selection for new users and is recommended where a user is unable to type their password consistently. A graphical model of the user's password performance (as shown in figure 3), recommendations and the ability to change the Biometrics Mode are detailed within a user's biometrics profile.

4.2 Active

Where the security of a user's login trumps the need for high availability an active approach to intrusion detection is preferred. The active action being the prevention of authentication this approach provides a functionality more appropriate in cases in contrast to passive mode. Currently active mode is available in three different levels of high, medium and low where high would constitute the acceptance of only very consistent keystroke data. This can be changed as different password policies and demographics could change the thresholds that should be used. Complexity requirements of a password could put a users typing behavior out of their comfort zone resulting in a increased inconsistency in their keystroke behavior. Additionally, a user base consisting of young adults versus senior citizens might see a different result as well.

5. ANALYZING BIOMETRICS DATA

As this project is a work in progress, analyzing the keystroke data that users provide is key in determining the proper thresholds and changes to deploy to ensure that the lowest error rates are achieved. The thresholds that determine what is flagged as the wrong keystroke behavior for a user is also very important to the administrators managing the system as the error rates may change depending on the password policy or user demographics. An administrator may also be able to discover additional information related to security given the proper tools to analyze the available data.

5.1 Detecting Multiple Account Creation for Single Users

Account systems can sometimes be subject to abuse by individuals who create several accounts to bypass usage policies. One method of detecting such an abuse could be to compare keystroke data between different accounts. Checking the Euclidean distance between each user's centroid can allow for possible duplicate accounts to be flagged or for immediate action to be taken by the system.

6. SECURITY CONCERNS

During development a few concerns became apparent in regards to possible attack vectors. For one, it should be noted that the keystroke data transmitted to a website is no more secure than the username and password from being intercepted by a third party. In this particular project the keystroke data is sent in clear text to the web server along with the username and password. It would therefore be highly advisable to use an encryption technology such as Transport Layer Security for transactions of authentication data. In the case that the keystroke data is compromised the attacker has the option of making an offline copy of the website so that the compromised data can be hard coded into the account form instead of letting the Silverlight control create the keystroke data.

Another concern lies with the storage of keystroke data. If the account database is compromised the keystroke data will be viewable in its plaintext format. In the case of the password however a hash is created and a salt is applied before being stored to at least increase the difficulty in obtaining the passwords. My current solution to this issue would be to separate the Keystroke Dynamics database from the account database. So if the account database is compromised the keystroke data will not be immediately accessible. Using a different database for keystroke data is possible as the Keystroke Dynamics tables do not have a key constraint with the account database.

7. FURTHER DEVELOPMENT

There is still much to do in developing an effective and fully functional implementation of Keystroke Dynamics. For one, there still exists the need to sample enough data to examine the error rates. There are also several improvements that can be made to increase the functionality of the system.

7.1 Effective Survey Development

There still needs to be a method to determine the false rejection and false acceptance rates. To do so there will need to be a modification to the bio_Datapoints table to allow the inclusion of failed keystroke data and an indication of whether a login was falsely accepted. The plan to accomplish this involves adding two fields into the bio_Datapoints table, one to indicate a failed login attempt and another to indicate if the login was done by the actual user. The failed login attempts can be logged by the account controller easily after failure to produce an acceptable distance from the users centroid. The indication of the user being genuine would need to be provided by the user before login such as by a checkbox that if checked would indicate the user is trying to log into someone elses account. This would be for data collection only and not for a system in production. A failed login by a user

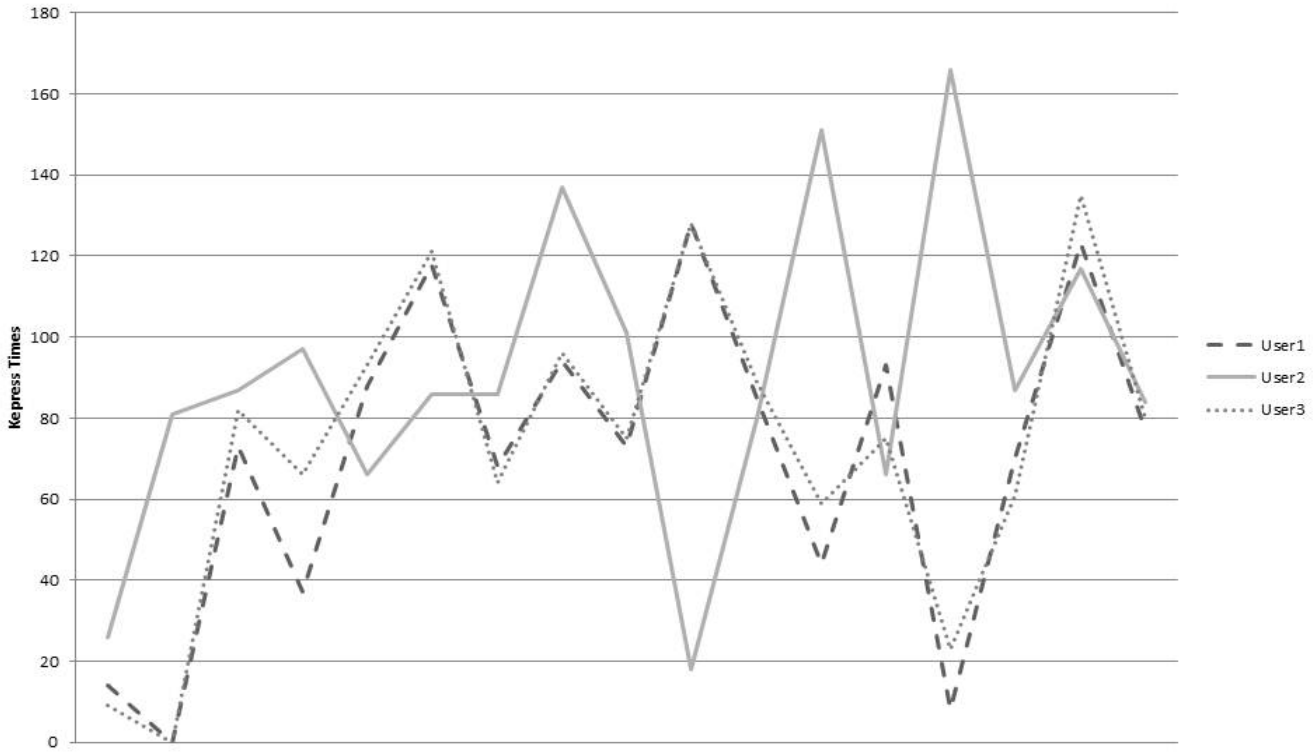

Figure 4: This chart shows the data attributes for three differenent user centroids. The short distance between User1 and User2 indicate a possible duplicate account.

with the right password and one who indicated that they are genuine would classify as a false rejection. Conversely, a successful login by a user who indicates they are not genuine would classify as a false acceptance.

7.2 Improvements to Functionality

The final goal in this project would be to produce a mature implementation of Keystroke Dynamics. The functionality, ease of use and even aesthetics are a key factor in the success of such a system. One improvement is to automate the process of recommending a biometrics mode to the user. There are several factors that can indicate whether an active or passive approach should be taken. Such factors could be the number of outliers, average distance from the centroid or to view the distance data as a normal distribution. Recommendations may also be deemed to be automatically set if the recommendation is sufficiently accurate. In some situations this may be the preferred method so as to increase the ease of use for users or to allow administrators to enforce a more stringent use of Keystroke Dynamics.

8. CONCLUSIONS

Although testing of the Keystroke Dynamics project has been limited, test subjects have reported the system's live demo to be easy to use and effective from a user standpoint. Unscientific as it be, I was still able to get a perfect success rate with "hack my account" games with fellow coworkers and friends. In this scenario I would give my username and password to another person to see if they could successfully authenticate. Even after others would watch me type in the password successfully and having the biometrics mode set to the lowest active threshold the results stayed the same.

If error rates can be reduced, at best this implementation of Keystroke Dynamics can be an affective addition to securing authentication from threats of stolen username and passwords. At worst it can work as an intrusion detection alert system to notify user and administrator of potential access exploitation.

9. REFERENCES

[1] K. S. Killourhy and R. A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *DSN*, pages 125–134, 2009.