Keystroke Dynamics using Auto Encoders

Yogesh Patel *(yop0043@my.londonmet.ac.uk)*, Prof. Karim Ouazzane *(k.ouazzane@londonmet.ac.uk)*, Dr. Vassil T. Vassilev *(v.vassilev@londonmet.ac.uk)*, Ibrahim Faruqi *(ibrahim.faruqi@callsign.com)*, and George L. Walker *(george.walker@callsign.com)*

***Abstract*—In the modern day and age, credential based authentication systems no longer provide the level of security that many organisations and their services require. The level of trust in passwords has plummeted in recent years, with waves of cyber attacks predicated on compromised and stolen credentials. This method of authentication is also heavily reliant on the individual user's choice of password. There is the potential to build levels of security on top of credential based authentication systems, using a risk based approach, which preserves the seamless authentication experience for the end user. One method of adding this security to a risk based authentication framework, is keystroke dynamics. Monitoring the behaviour of the users and how they type, produces a type of digital signature which is unique to that individual. Learning this behaviour allows dynamic flags to be applied to anomalous typing patterns that are produced by attackers using stolen credentials, as a potential risk of fraud. Methods from statistics and machine learning have been explored to try and implement such solutions. This paper will look at an Autoencoder model for learning the keystroke dynamics of specific users. The results from this paper show an improvement over the traditional tried and tested statistical approaches with an Equal Error Rate of 6.51%, with the additional benefits of relatively low training times and less reliance on feature engineering.**

***Index Terms*—Keystrokes Dynamics, Keystroke Analysis, Autoencoders.**

I. Introduction

THERE is a rise in the field of computer technology that has brought a lot of insecurity, such as credential based attacks. Today user ID and password are widely used methods of authentication to verify user identity. The notion of password dates back to early 1960s, when Massachusetts Institute of Technology produced a system called Compatible Time Sharing System (CTSS) [21]. Initial theory was that passwords provide "something you know", which is easy to remember and harder to guess by adversaries. However, in practice users will commonly share passwords, use the same password across multiple sites or will pick weak credentials that are easy to guess or reverse engineer. Exploitation of these flaws has resulted in the proliferation of threat vectors such as masquerading and identity theft attacks.

These attacks are highlighted in several major studies:

- According to the World Economic Forum [19], in 2017, the cost of cybercrime to businesses was more then $500 billion where different business verticals ranging from government to utility services and financial services were attacked.
- A recent Verizon Data Breaches Report [7] states that the majority of data breaches (2 out of 3) were a direct result of compromised passwords.
- Earlier in 2016 [20], a survey by MasterCard suggested that more then 50% of users forget passwords more than once a week, resulting in high abandonment rates and poor customer experience.

The above facts suggest that the traditional password based authentication scheme is insecure, costly and inconvenient. There is a need to provide an additional layer of security control to discriminate between the genuine user and an imposter, without impacting customer experience.

An increasingly popular method to achieve this is via the use of behavioural biometrics. This is the process of measuring the unique patterns of human activity to learn the typical behaviour of a user so as to prevent adversarial attacks. This is performed by converting gathered biometric data into a numeric value and comparing this mathematically to a dataset of normal behaviour for the user. Examples of behavioural biometrics include speech, mouse, touch and signature verification capabilities.

Keystroke dynamics is another behavioural biometrics capability that captures the typing rhythms of the genuine user and rejects authentication attempts that deviate away from these. This can further be used to determine the difference between a human typing or scripted programs such as malware [1]. One of the major benefits of keystroke dynamics is that it can be deployed as a covert operation and does not require any special hardware, this makes it an attractive technique for defending the cyberspace [2]. Yu and Cho [22] note that keystroke dynamics is a more effective solution when compared to use of passwords alone, since they are easily compromised.

Keystroke dynamics is a well researched topic and several algorithms have been proposed for detecting anomalous behaviour and/or imposters. The research by Jiaju H, et al. [9] summarises the latest comparison of the algorithms used and suggests further benchmarking is required due to the lack of an unconstrained dataset.

Several existing algorithms make use of features such as dwell time and flight time [10] to create a user template that encodes an individual's keystroke dynamics. However, in the real world, noise introduced by modifier keys such as deletions and substitutions of the keystroke events creates inconsistencies in the template creation process. Thus, making it harder for algorithms to discriminate between genuine and imposter. In order to compensate for such discrepancies, a robust algorithm is required that can maximise the amount of data available for training and testing to achieve better performance.

This paper proposes an Autoencoder as an anomaly detection technique that can be used to establish a semantic correlation between the keystrokes template and an approximately

matching query. An Autoencoder reconstructs the inputs as outputs by compressing the representation via an intermediate layer. The hypothesis is that the learnt representations accurately encode features for the legitimate user with minimal reconstruction error, but are less efficient at encoding features for imposter users, represented as high reconstruction error. The rest of this paper is organised as follows: Section II background and related work. Section III describes the model architecture. Section IV describes the performance evaluation and results. Section V provides conclusion and a discussion.

II. Background and Related Work

A. Background

The first use of keystroke dynamics in information technology (IT) appeared in Spillance, R.j. in 1975 and made use of keystroke timings to identify an individual user [18]. As discussed in the introduction, keystroke dynamics captures the unique characteristics that exist in individual's typing behaviour. According to Spillance, a keystroke dynamics template can be created by tracing the timings that are represented as a series of digraphs. A digraph captures the timing for two adjacent keystrokes and provides a set of features as described below [21]:

- Dwell Time: Time spend on the actual key
- Flight Time: Time taken to move from one key to the subsequent key.

These features can then be concatenated into a vector and used as a sequence of keystroke timings, which encapsulates the behaviour of the user typing their credentials. Other potential keystroke features include keydown-keydown and keyup-keyup times. Trigraphs are also studied in the literature, which are the time latencies between every three consecutive key down press and similarly, n-graphs [30]. Flight and dwell times are highlighted in Figure 1:

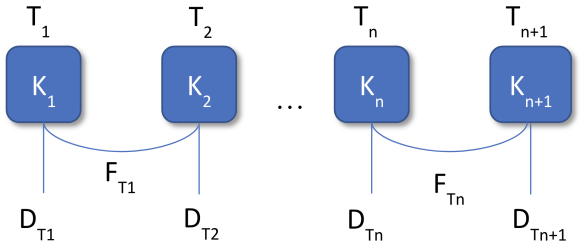

Figure 1. Demonstration of Dwell Time and Flight Time

In the above Figure 1, T_n represents the time at which an event took place, K_n is the key pressed at time T_n, D_{Tn} is the dwell time for the key K_n and F_{Tn} is the flight time between keys K_n and K_{n+1}. Each key press event can be stored in a database as a set of the timing features based on the keystroke press/release timestamps. There are no standard based approaches to represent this data, but most research and literature utilises these features.

Performance Stats		
Authors	FAR	FRR
Legget & Williams (1980) [23]	0%	4%
Joyce & Gupta (1990) [24]	0.25%	16.74%
Bleha et al. (1990) [25]	2.8%	8.1%
Gunetti & Picardi (2005) [26]	0.5%	5.1%
Ahmed & Traore (2008) [27]	0.0152%	4.82%
Kang et al. (2007) [28]	3.8%	3.8%
Yu & Cho (2003) [22]	0.0%	15.78%
Araujo et al. (2004) [29]	1.89%	1.45%

Table I
Investigation of different authors and the performance.

B. Related Work

Establishing identity using the keyboard characteristics was proposed in the RAND report [6], funded by the National Science Foundation. They used a digraph representation for the keystrokes and conducted experiments on a small population of users. Keystroke dynamics relies on recognition of patterns in the user typing, as such, its success depends on correlation based feature selection as suggested by Mark, et al. [14].

Several research papers [2], [9] attempt to create a benchmark and compare keystroke dynamics algorithms. Their reports seem to suggest that distance based algorithms (utilising metrics such as Manhattan and Euclidean) and Neural Networks achieve reasonable amount of success, but are prone to noise and outliers in genuine attempts. One particular study highlighted in [2] by Gunetti and Picardi uses n-graph flight times and sum of degree of disorder, which seems to achieve False Acceptance Rate (FAR) of 0.5% and False Rejection Rate (FRR) of 5%. However, for this algorithm to be effective, information on the actual values of the keys pressed is required, this may cause data privacy and security issues when applied to credentials. A summary of the existing research and results are detailed in Table I, but the diversity of the datasets and evaluation criteria makes the results hard to interpret and benchmark.

Hosseinzadeh [8] suggested the creation of a keystroke dormancy feature and related its performance with existing features using a Gaussian Mixture Model (GMM) based verification system. The results proved that the Up-Up Keystroke Latency (UUKL) feature significantly outperformed the commonly used key hold-down time and down-down keystroke latency features. Nevertheless, as the length of the text increases, the change in the discriminability between the assorted and aggregate vectors tends to decrease. Kevin [11] merged keystroke dwell time for user authentication built on the keystroke dynamics of the password entry. Instead of using the complete dataset for training, only the keystroke dynamics of a small subset of users, referred to as representatives, was used along with the password entry keystroke dynamics of the examined user. By doing this, the risk of over fitting is reduced, while allowing scalability to a high volume of users.

Pawel K, et al. [10] makes use of the Recurrent Neural

Network topology, Long Short Term Memory (LSTM), to treat keystroke events as a sequence to create a robust predictor that can deal with modifier keys. However, such topology requires a large amount of data and a lengthy training period which may be difficult to obtain for an individual user in practice.

Keystroke dynamics have also been applied to the automated teller machines (ATMs) [15] where users have been asked to enter a Personal Identification Number (PIN). However studies suggest that numeric keypads differ significantly from keyboards as the use of the keypads are based on a single finger and hand which makes it harder to determine individuals due to less neuromuscular entropy. Several algorithms, such as Neural Networks and minimum distance based, were applied by Ord, et al. [16] on a small dataset of 14 users entering a common six-digit PIN and found Neural Networks outperformed other algorithms but still had a FAR of 9.9% and FRR of 30%.

III. Model Architecture

The model proposed in this paper aims to verify users based on their keystroke dynamics using features suggested in Section II. Verification differs from identification in that classification is binary, the aim is therefore to find whether a user is who they claim to be rather than discover the user's identity from a population. To achieve this a keystroke dynamics dataset is required which can be subset into imposter and genuine attempts. A statistical model must then be implemented which can provide an authenticity score for each attempt and have overall performance measured using Equal Error Rate (EER).

The system is therefore broken into two major bodies of work:

- Data Acquisition: Firstly, several publicly available datasets were examined and the Carnegie Mellon University (CMU) dataset was chosen, as detailed in Figure 2. Keystroke timing data was extracted from the dataset along with individual user ids.
- Model Processing: Responsible for extracting relevant features, executing a model that will; a) create the behaviour template during the training b) create the matching query during testing and prediction The output of a prediction will be a similarity score that will be normalised. A threshold will be applied to distinguish between genuine and imposter attempts.

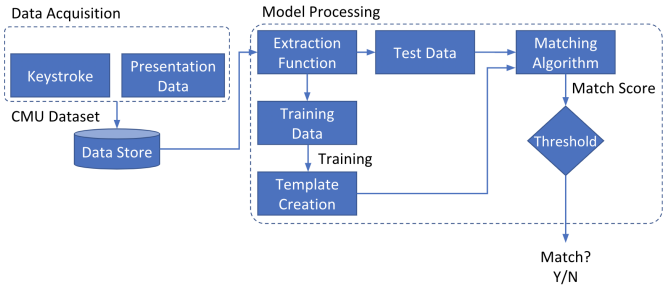

Figure 2. Keystroke System Component Diagram

A. Data Acquisition

For the purpose of this paper the authors have used the CMU's benchmark keystroke dataset proposed by Kevin S., et al. [12]. This dataset makes the comparison easier for benchmarking. For example, the paper by Kevin highlights many anomaly detectors such as Manhattan and Mahalanobis achieving reasonable accuracy with EER of 9.6% and 10.0% respectively. The data consist of keystroke-timing information from 51 users, each typing a password (.tie5Roanl) 400 times. Since, all users type the same password, test and training sets can be easily partitioned, where negative data/imposter attempts can be sampled from the other users.

B. Model Processing

The primary goal for model processing is to create an environment where models such as Autoencoders and Gaussian Mixture Models can be implemented for anomaly detection. As shown in Figure 2, this involves feature extraction(where necessary) and splitting of the data into training and testing sets, for validation of the model performance.The hyper parameters for each of the models were selected by evaluating their performance on a small development set.

1) Autoencoder: Autoencoders are widely used today. They work by mapping the output to the input representations with minimal amount of distortion. An architecture of the typical Autoencoder is similar to the Multi Layer Perceptron (MLP) i.e. an input layer, one or more hidden layer(s) and an output layer. An output layer has the same number of neurons as the input layer. The architecture of an Autoencoder consists of:

- an encoder layer that provides a deterministic function to map the input vector x into a hidden representation called z. The deterministic function is given by:

$$f(x) = s(Wx + b) \tag{1}$$

  Where, W is the weight matrix, b is the bias vector, s is the non-linear activation function such as $tanh, sigmoid, ReLU$.
- a decoder layer takes the hidden representation z and outputs a reconstructed version of the input x. A typical non-linear squashing function is given by:

$$g(z) = s(W'z + b') \tag{2}$$

In general, the predicted output, will not be an exact representation of input x but rather a probabilistic output, that defines a distribution $p(X|Y = y)$ and hence, able to define our reconstruction error that needs to be optimized as:

$$L(X, Y) \propto -logp(x|y) \tag{3}$$

Assuming the keystroke input as a Gaussian representation, we can apply an $L2$ norm to derive:

$$L(x, y) = L_2(x, y) = ||x - y||_2^2 \tag{4}$$

An overall objective function is given by:

$$Q_t(\theta) = \sum_{x \in D} L(x, g(f(x))) \tag{5}$$

Where, D is the training samples. The weight matrix W and the bias vector b are initialised from a random uniform distribution such that the scale of the gradients in each layer is roughly the same.

Figure 3 illustrates the layout of the Autoencoders used in this work, which is encoding and decoding the keystroke sequences (during the testing phases, several additional hidden layers may be added in order to fine tune model performance). The individual input consists of a single attempt and the output is a reconstruction of the attempt. Training will be performed in order to minimise the reconstruction errors by maximising the lower bound on entropy between input X and reconstructed output Y.

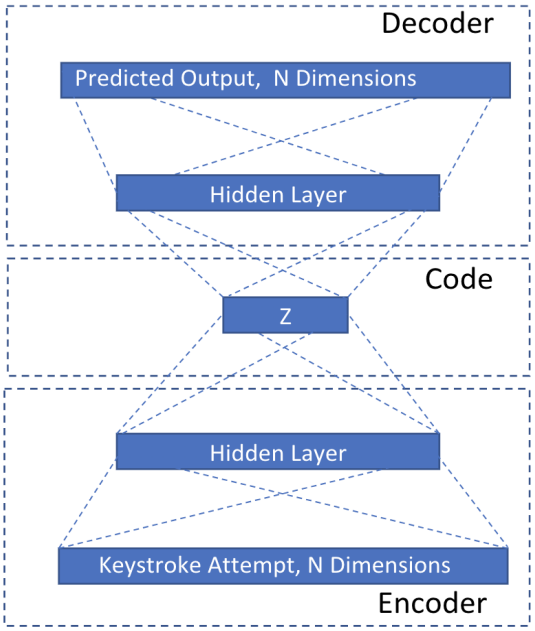

Figure 3. Autoencoder for keystroke sequence

The extracted features from the initial layer are the encoded representation of the input, that the Autoencoder learns during training. The encoded samples are used to train a classifier, either using only the encoded representation or in combinations with other features, such as dwell time, flight time and digraph. Tests were performed using Autoencoders, with several different configurations, with the goal of learning how an Autoencoder can be built and how it performs. Different optimizers were tested in order to find a suitable method for training an Autoencoder: Adam, Stochastic Gradient Descent, Adadelta and RMSprop. Initially ReLU was used as the activation function for the hidden layers. Later a Sigmoid was chosen for the encoding layer as it resulted in a slight improvement of the reconstruction error. ReLU is used as the activation function for the other hidden layers.

IV. Experiments, Results & Discussion

The CMU Dataset[1] contains 20400 records with 34 columns. It has 50 subjects and each subject has 400 password entries. The evaluation will be based on the method described by Steven F [4] where 80% of data will be used for training (320 entries per user) and 20% will be used for testing (80 entries per user) and validation of model performance against positive data. Since, all subjects entered the same text, imposter attempts can be simulated by sending a random user's data to contrast the similarity score. To generate negative data, for each subject, a random user will be selected from the remaining population (20% of the selected user's samples will be used to represent imposter attempts). This data will be used to test and validate the model's performance against potential fraud attacks utilising stolen credentials.

In order to baseline the dataset and to validate the output for correctness, the Euclidean distance from the centroid of the training data will be used. To quantify performance, EER is calculated. The experiments were repeated several times and the mean EER is calculated. The mean EER is given as 0.19 with a standard deviation of 0.09. As observed, the obtained values closely matches the value reported by Kevin, et al. [11] for the 'Euclidean' method.

A. Exploratory analysis

In order to explore the marginal distribution of each variable in the dataset, several random subjects' data were taken and the Kernel Density Estimation (KDE) [3] technique was applied. Figure 4 provides an output of mean distribution of 5 different subjects.

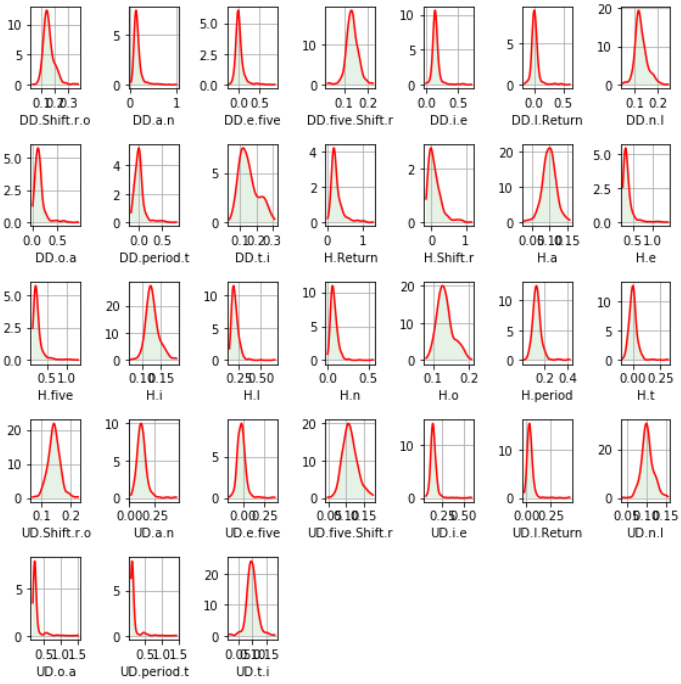

Figure 4. KDE of keystroke timing features for different users. DD refers to the key down-down latency, H refers to the hold time (dwell time) of the key and UD is the up-down latency (flight time). The additional information details the key/key combination for which these timings were observed

According to Figure 4, the majority of the marginal distributions are asymetric and unimodal. Marginal distributions such as 'H.Return', 'H.Shift.r' appear to be relatively skewed.

[1]Keystroke Dynamics - Benchmark Data Set (https://www.cs.cmu.edu/ keystroke/)

Feature Transformation	Equal Error Rate (ERR)
PCA ($ngaussians = 1$, $ncomponents = 35$)	0.1028
KPCA ($ngaussians = 1$, $ncomponents = 20$)	0.093

Table II
GAUSSIAN MIXTURE MODEL WITH FEATURE TRANSFORMATION USING PCA AND KPCA

Whereas outliers appear to be present in some distributions (e.g. 'H.t', 'DD.period.t'). Based on visual inspection they appear to be within one order of magnitude relative to the standard deviation, hence relatively small.

For a few users, the data was split into training and testing sets (80:20 split). The training set was used to plot the KDE of the genuine user, which was then overlaid with the KDE for imposter data. This is shown in Figure 5, where a clear split in the density between genuine and imposter user behavior is observed.

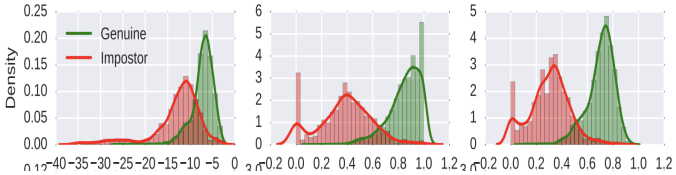

Figure 5. Applying Density Estimation to genuine and imposter data

B. Gaussian Mixture Model

In order to compare the results from the Autoencoder, a Gaussian Mixture Model is used to obtain an anomaly score for a given subject using the estimated probability density associated with the instance. As independent variables, the number of Gaussians (ranging from 1 to 7), and the type of covariance matrix estimated for each Gaussian (full or diagonal) were considered. Since assuming diagonal covariance for each Gaussian potentially discards information about correlated features [17], a preprocessing step was investigated to evaluate the effects of decorrelating variables. This was done by applying Principal Component Analysis (PCA) and Kernel Principal Component Analysis (KPCA).

Figures 6 and 7 highlight EER plots for each Gaussian covariance type / feature transformer combination. For the case where no feature transformations are applied, the mean EER is plotted for each of the number of Gaussians that were investigated. For the cases where PCA or KPCA is applied, the mean EER is plotted against the number of components that were used in the PCA/KPCA, for each of the number of Gaussians that were investigated.

A summary of the EER results are described in Table II:

C. Autoencoder

The implementation of the Autoencoders consists of two encoder layers and two decoder layers, training is performed using the ADAM [13] optimiser and the loss is calculated by

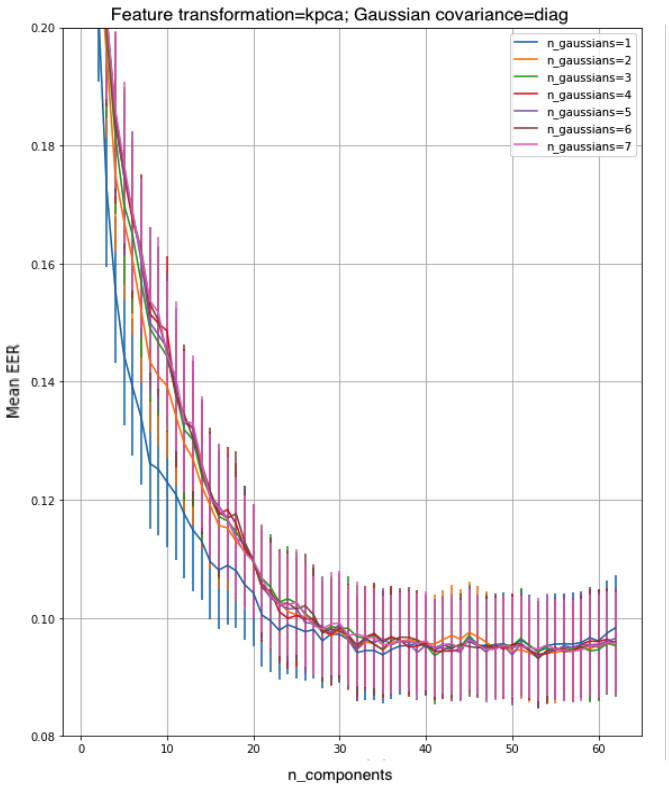

Figure 6. Gaussian Mixture Model with Principle Component Analysis

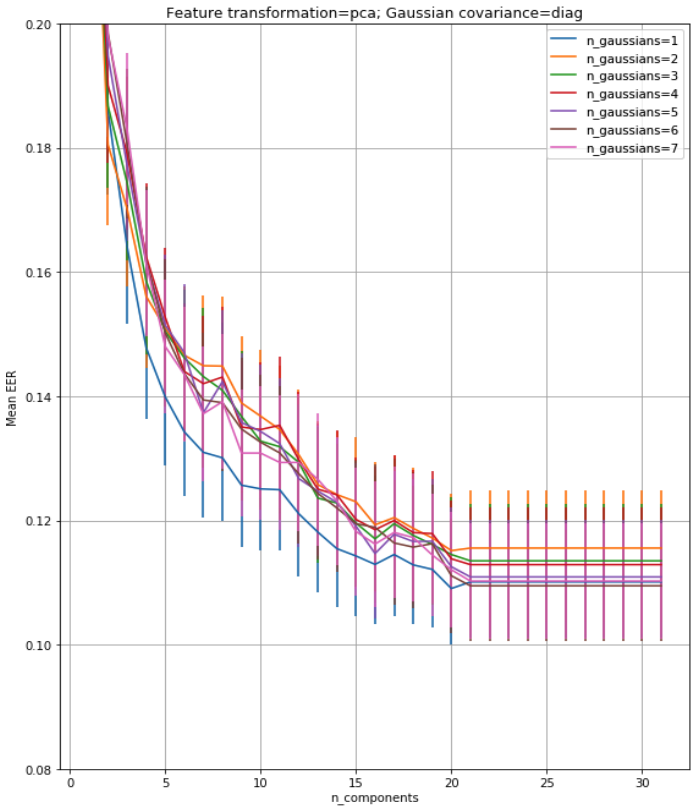

Figure 7. Gaussian Mixture Model with Kernal Principle Component Analysis

Network Hyperparameter	
Parameter	Values
Activation Function	ReLU
Dropouts for Encoding	20%, 10%, 5%
No. of Epochs	50
Learning Rate	0.0001
Weight Decay	0.001

Table III
FINAL NETWORK PARAMETERS FOR AUTOENCODER

optimising the mean squared error of the reconstructed input. To aid convergence and to quantify the similarity of different samples, data normalisation is applied to rescale each sample's features to have zero mean and unit variance. To help prevent the Autoencoder from learning the identity function, a sparsity penalty is applied on the hidden activations using different dropouts for each layer. After several trial and error runs, the final list of hyperparameters used are shown in Table III.

Several runs were performed using the hyperparameters highlighted in Table III and mean values are calculated to represent the final evaluation. Using the above parameters the model is trained on the individual subject using the 80% of the samples, 20% are held back and used for testing. The mean training losses for different activation functions are presented in Figure 8.

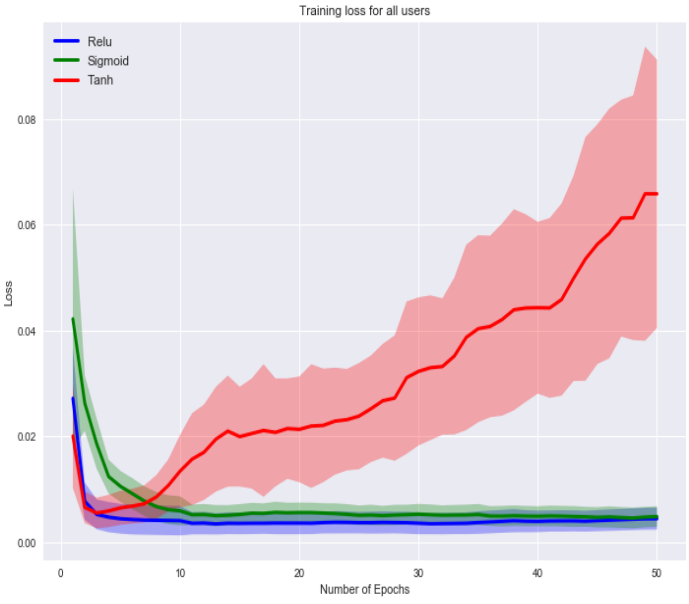

Figure 8. Training using Sigmoid, ReLU and Tanh Activation Function

The training highlights that the ReLU converges quicker than other two activation functions. Once the model is trained on individual users, the 20% of the testing data will be used to evaluate the performance along with that imposter dataset. The Mean Squared Error (MSE) across all users and the imposters is shown in Figure 9.

From Figure 8, the performance of different activation functions can be compared. Using tanh resulted in poor performance, with the model not converging. Once the model performance begins to diverge it is unable to recover and continues to produce poor results. The performance of tanh could have been improved by tuning the hyperparameters such as the learning rate and weight decay. We observe a noticeable benefit of using ReLU over Sigmoid, as the model learns the user's behaviour much faster. The loss was significantly lower when using ReLU, especially for a low number of Epochs (less than 10). This is particularly useful in the instance where regular retraining of models is required to adapt to changes and quirks in user behaviour over time. However, it is observed that both ReLU and Sigmoid eventually converge to have relatively the same performance, as the number of epochs is increased. The mean EER for each activation function is shown in Table IV.

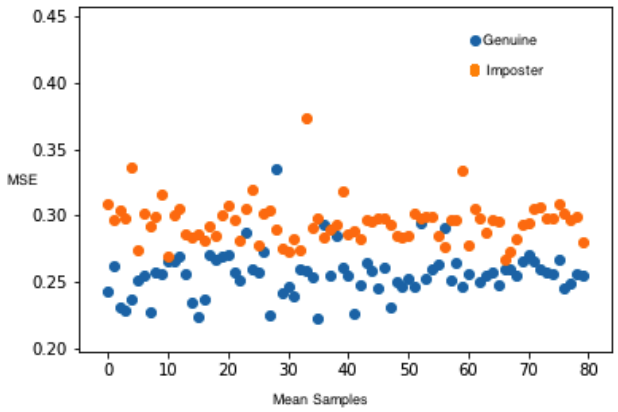

Figure 9. Mean Square Errors between users and imposters

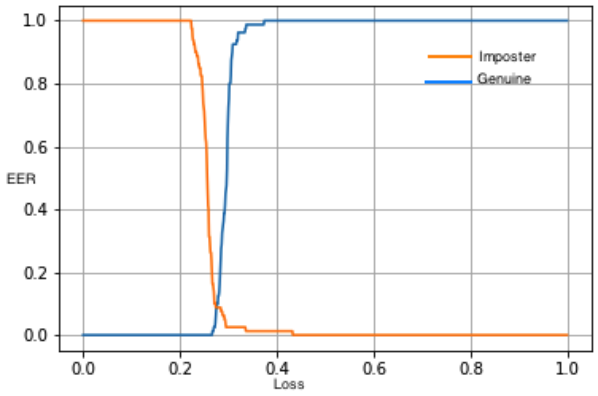

Figure 10. Mean Equal Error Rate

Figure 9 highlights the proof of the original hypothesis: that Autoencoders can discriminate between genuine and imposter users, through the reconstruction error. The figure shows a clear divide in the MSE between genuine and imposter users, with imposters having a noticeably higher MSE. This is further highlighted in Figure 10, where the EER is plotted against the loss values. The best threshold value for the loss (to distinguish

Activation	Regularisation	EER
ReLU	None	9.23%
ReLU	Dropout	6.51%
Sigmoid	None	14.526%
Tanh	None	23.742%

Table IV
EER vs Activation Functions

between genuine and imposter) can be observed visually by looking at the point of intersection between the genuine and imposter graphs. The optimal threshold in this research, was found at a value of 0.28.

This paper evaluates two anomaly detection approaches, namely Gaussian Mixture Models and Autoencoders, against the CMU keystroke dynamics dataset. Based on reported EERs in Tables II and IV and the considered methodology, the best-performing approach yields an EER of 6.51%, obtained using an Autoencoder with no feature transformation applied besides standardisation. For the Autoencoder, dropout was used as a regularisation method to increase the discriminative performance and it was observed that EER was further dropped by 2.72%. It was also noted that the multi-layered Autoencoder performed roughly the same as the single-layered Autoencoder, even though it has a greater representational potential capacity since it has more nodes. This might be due to the lack of variance in the data, as well as the limited size of the CMU dataset, hence the absence of any performance benefit from using additional layers. Using GMM with KPCA in combination with KDE yields performance of 9% EER as shown in Figure 7.

V. Conclusion

As this paper demonstrates it is possible to create a risk based authentication system that can utilize behavioural biometrics such as keystroke analysis. The results highlighted, that by utilizing a state of the art algorithm such as an Autoencoder, it is possible to sufficiently compete with the baseline performance of traditional statistical and machine learning methods. Furthermore, it is possible to further achieve performance improvement by utilizing techniques such as regularisation. The advantage of Autoencoders, in the minimal feature engineering that needs to be performed, is also highlighted along with the relatively quick training times. Since one of the benefits of keystroke dynamics as a risk based authenticator, is to minimise the impact on the end user's experience, needing less feature engineering means less time being spent on data processing. This in turn can reduce the total time taken to assess the risk of the login based on the keystroke behaviour.

Based on the EER values obtained in this study, it can be said that keystroke behavioural biometrics has the potential to play an important role in creating risk based authentication systems, however they cannot be relied on in silo'ed. In order to create a more robust system with lower error rates, it needs to be augmented with other risk factors by ensembling other biometric traits such as mouse movement. Moreover, initial exploration suggests that there is a learning effect in subjects' data so there might be another possibility to boost performance using a time varying sampling technique, e.g. by filtering across time. In future work, hyperparameters can be evaluated more closely and a more extensive grid search can be performed for Autoencoders.

In general, as the users move more towards mobile and mobile application based usage, there will be a limited amount of keystroke data that will be available, which will make behaviour biometrics using keystrokes more difficult. Also, users may utilise Password Managers or use other techniques such as copy and paste, which may render keystroke biometrics as a less effective indicator of risk.

In addition future work could apply Autoencoders to a dataset which is more representative of a user's behaviour, when interacting with an application or service. The CMU dataset was gathered in a controlled environment, with users typing in credentials consecutively, which may produce more consistent typing patterns than is normal for end users. Further study could also be augmented by the addition of extra data, in the form of mouse behaviour data and (where possible) touchscreen events, where the relationship between these interactions and keystrokes can be investigated.

References

[1] Rafay Baloch. An introduction to keyloggers, rats and malware.

[2] S. P. Banerjee and D. Woodard. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research*, 7(1), 2012.

[3] Yen-Chi Chen. A tutorial on kernel density estimation and recent advances. *Department of Statistics, University of Washington*, 2017.

[4] S. Finlay. *Predictive Analytics, Data Mining and Big Data: Myths, Misconceptions and Methods (Business in the Digital Economy)*. Palgrave Macmillan, 2014.

[5] Europen Committee for Electrotechnical Standardization. European standard en 50133-1: Alarm systems. access control systems for use in security application. *CLC/TC79*, 2002.

[6] R. Stockton Gaines, William Lisowski, S. James Press, and Norman Shapiro. Authentication by keystroke timing - some preliminary results. 1980.

[7] Verizon Group. 2017 data breach investigation report.

[8] D. Hosseinzadeh and S. Krishnan. Gaussian mixture modeling of keystroke patterns for biometric applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(6), 2008.

[9] Jiaju Huang, Daqing Hou, Stephanie Schuckers, Timothy Law, and Adam Shrewin. Benchmarking keystroke authentication algorithms. *IEEE - Information Forensics and Security (WIFS)*, 2017.

[10] Pawel K. and Khalid S. Application of recurrent neural networks for user verification based on keystroke dynamics. *Journal of Telecommunications and Information Technology*, 2016.

[11] Kevin S. Killourhy. *A Scientific Understanding of Keystroke Dynamics*. School of Computer Science, Carnegie Mellon University, 2012.

[12] Kevin S. Killourhy and Roy A. Maxion. Comparing anomaly detectors for keystroke dynamics. *IEEE - Annual International Conference on Dependable Systems and Networks*, 39, 2009.

[13] Diederik P. Kingma. Benchmarking keystroke authentication algorithms. *IEEE - Information Forensics and Security (WIFS)*, 2017.

[14] H Mark. Correlation-based feature selection for machine learning. *Department of Computer Science*, 5(23), 2000.

[15] Akio Ogihar, Hiroyuki Matsumura, and Akira Shiozaki. Biometric verification using keystroke motion and key press timing for atm user authentication. *Intelligent Signal Processing and Communications*, 2006.

[16] T Ord and S.M. Furnell. User authentication for keyboard-based devices using keystroke analysis. *ISpinnaker International Ltd., Plymouth, UK*, 1999.

[17] M. A. Osborne, S. J. Roberts, A. Rogers, S. D. Ramchurn, and N. R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. *International Conference on Information Processing in Sensor Networks*, 2008.

[18] Spillane R.J. Keyboard appratus for personal identification. *IBM Technical Disclosure Bulletin*, 17(3346), 2012.

[19] Victoria Golshani Shirazi. Lessons from the latest wave of cyber-attacks.

[20] Jennifer Stalzer. Mastercard identity check to simplify and strengthen online shopping.

[21] John D. Woodward, Nicholas M. Orlans, and Peter T. Higgins. *Identity Assurance int he Information Age - Biometrics*. McGrawHill Osborne, 2003.

[22] E. Yu and S. Cho. Keystroke dynamics identity verification-its problems and practical solutions. *Computers & Security*, 5(23), 2004.

[23] J. Leggett and G. Williams. Verifying identity via keystroke characteristics. International Journal of Man-Machine Studies, 1988.

[24] , R. Joyce and G. Gupta. Identity authentication based on keystroke latencies. Communication of the ACM, 1990.

[25] S. Bleha and C. Slivinsky and B. Hussien. Computer-access security systems using keystroke dynamics. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990.

[26] C. Picardi and D. Gunetti. Keystroke analysis of free text. ACM Transactions on Information and System Security, 2005.

[27] A. Ahmed and I. Traore and A. Almulhem. Digital Fingerprinting Based on Keystroke Dynamics. Proceedings of the Second International Symposium on Human Aspects of Information Security and Assurance, 2008.

[28] P. Kang and S. Hwang and S. Cho. Continual retraining of keystroke dynamics based authenticator. In Proceedings of the 2nd International Conference on Biometrics, 2007.

[29] L. Araujo and L. Sucupira and M. Lizarraga and L. Ling and J. Yabu-uti. User authentication through typing biometrics features. In Proceedings of the 1st International Conference on Biometric Authentication, 2004.

[30] Y. Zhong and Y. Deng. Keystroke Dynamics User Authentication Using Advanced Machine Learning Methods. Recent Advances in User Authentication Using Keystroke Dynamics Biometrics. Vol. 2, pp. 23-40, 2015.