2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Two-Factor User Authentication with the CogRAM Weightless Neural Net

Weng Kin Lai, Beng Ghee Tan, Ming Siong Soo and Imran Khan

***Abstract*— The application of the *Cognitive RAM* (CogRAM) weightless neural net in testing a keystroke biometrics user authentication system for a numeric keypad is discussed in this paper. The two-factor user authentication system developed here uses the common password that is complemented with the keystroke patterns of the users. The keystroke pattern is represented by the force applied to constitute a fixed length passkey to compose a complete pattern for the entered password. The system has been designed and developed around an 8 -bit microcontroller, based on the AVR enhanced RISC architecture. The preliminary experimental results showed that the designed system can successfully authenticate the unique and consistent keystroke biometric patterns of the users.**

I. INTRODUCTION

In the context of biometrics, *recognition* and *authentication* are two common but different applications of biometric technologies. Recognition involves finding the one unique identity amongst the many stored identities, whereas authentication relates to matching or verifying the patterns against a single user's stored identity. Essentially, recognition is the claiming of an identity whereas authentication is the act of verifying or proving the claimed identity.

Although a variety of authentication devices may be used to verify a user's identity, passwords remain the most preferred method especially when a keyboard is the data entry device. This is because password authentication is relatively inexpensive, intuitively familiar to most users, and supported by most operating systems. However, unless it is used correctly, the level of security provided by passwords can be low. Despite many years of widespread use, the issue of weak user password still exits. Hence, multi-factor approaches are needed to extend and strengthen the security level that passwords provide. This reinforcement should be transparent and indiscernible to the users and does not require any additional efforts while they are entering the normal authentication information (user ID and password). In addition to different and personalized passwords for each user, the users also have a unique way of using the keyboard to enter their passwords. For example, each user may type the characters that constitute the password at different speeds. By leveraging on these differences, one can develop a methodology that may be used to improve security by using keystroke biometrics (or in some literature, typing biometrics) to reinforce password-authentication mechanisms. Previous research [1-4] has shown that it is possible to identify a user via his or her typing patterns. Keystroke biometrics is the analysis of a user's keystroke patterns. An individual's keystroke biometrics pattern can be based on any combination of the following features[3] [5],

(a) *the duration each keystroke is pressed, that is the amount of time a user takes to press and release when typing*,
(b) *the duration between each pair of keystrokes*,
(c) *the force exerted on the keys*.

Our current research focuses on using the force exerted on each button to create the individual and personalised typing pattern. Based on the typing patterns obtained, the keystroke biometrics methodology first computes a typing template for the user. The authentication system would then save this against the associated identity along with the password. On subsequent attempts to access the system, the user goes through the normal password authentication procedure that is, entering the user ID and password. At the same time, the system monitors the user typing patterns and computes a typing template based on the user's ID and password just entered and compares this template with the stored template for this user. If the new password and typing template match those saved in the database, the system grants access to the user. However, if the password does not match, the normal password-authentication mechanism (without consulting the biometrics component) will reject the user or ask the user to reenter the authentication information. If the password does match, the biometrics system will provide a supporting recommendation that verifies whether the user is legitimate. If the user ID and password are correct, but the new typing template does not match the reference template, the security system has several options, which may be devised accordingly.

In this study, we will be investigating how we may authenticate an individual's keystroke biometric pattern based on the force (or amount of pressure) exerted on each key using a weightless neural network (WNN). The hardware design has been implemented to achieve desired results with minimal hardware component utilization. The biometric sensors of the system are force sensitive sensors which were used to translate the amount of force exerted on the keys into their equivalent electrical values, so as to give an accurate representation of the amount of force that each user applies while typing.

Weng Kin Lai is with the Electrical & Electronic Engineering Department, Faculty of Engineering & Built Environment, Tunku Abdul Rahman University College, Malaysia. (corresponding author: +603 4145 0123; fax: +603-4142 3166; e-mail: *laiwk@acd.tarc.edu.my*)

Beng Ghee Tan and *Ming Siong Soo* are with the Mechanical Engineering Department, Faculty of Engineering & Built Environment, Tunku Abdul Rahman University College, Malaysia.

Imran Khan is with the Dept. of Electrical and Computer Engineering, IIUM, Malaysia.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

Two main authentication issues are emphasized during the overall design of the system, viz.

(a) the numeric password representing the normal passkey entered by the user, consisting of numeric combination of the appropriate length created by the user and saved by the system.

(b) the keystroke biometrics associated with the user's password in the form of a "*typing template*". This is the second factor which will be authenticated by the weightless neural net.

Some prior work has been done on the use of keystroke biometrics as a password hardening technique[6-9]. The results reported of authenticating users based on just their keystroke have been encouraging. Nonetheless, their work has been centered primarily on the common QWERTY computer keyboard. While the QWERTY layout has been used extensively in the past, the simple and inexpensive numeric keypad is gaining popularity. However, the typing style is significantly different for the QWERTY keyboard when compared with that of a numeric keypad. Fig. 1 shows a common QWERTY computer keyboard on the left and a numeric keypad on the right.

(a) QWERTY keyboard (b) Numeric keypad

Fig.1. QWERTY keyboard and a numeric keypad

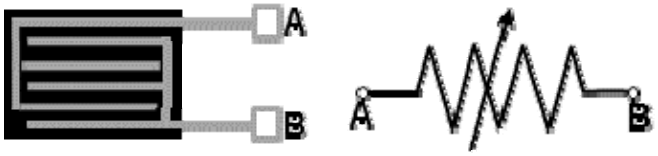
In addition to the duration between each pair of keystrokes, *Obaidat and Sadoun* [10] investigated the use of the holding time for each key pressed, using a QWERTY computer keyboard.

The remainder of this paper is organised as follows. The next section discusses the hardware design of the keystroke biometrics user authentication system which examines the force exerted by users on a numeric keypad. In section III, we describe the important details of weightless neural networks (WNN's) for pattern recognition, with a brief description of related work. It also introduces the *Cognitive RAM* network (CogRAM) and details its architecture and learning rules. Section IV discusses the major design issues as well as the important aspects of the experiments and the results obtained. Finally, in section V we present some conclusions and potential areas for further work.

II. HARDWARE DESIGN

A *Force sensor*

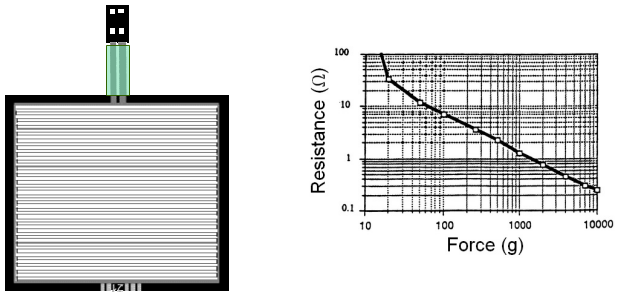
Force sensitive resistors are usually made from a conductive polymer that changes resistance in a predictable manner following application of force to its surface[11]. It is made up of two parts. The first is a resistive material applied to a film. The second is a set of digitating contacts applied to another film. Fig. 2 shows this configuration.



(a) Force sensitive resistor (b) Schematic representation

Fig.2. Force sensitive resistor

The resistive material serves to make an electrical path between the two sets of separated conductors. When a force is applied to this sensor, the resistivity between the contacts drops and the overall conductivity increases. Over a wide range of forces, it turns out that the conductivity is approximately a log-linear function of force ($F \propto C^{\alpha}$, $F \propto \frac{1}{R^{\alpha}}$), where α represents the sensitivity and linearity of the sensor. The FSR used in the user authentication system developed here has a sensing area of 1,75 inches × 1.50 inches, as shown in Fig. 3(a). When no force is applied, the resistance will be about 1 MΩ and it can detect forces from a mere 100 g up to 10 kg. Fig. 3(b) shows the resistance of the sensor as a function of force. It is important to note that there are three regions of operation of the sensor. The first is the abrupt transition which occurs somewhere in the vicinity of 10 grams of force. In this region the resistance changes very rapidly. Above this region, the force is approximately proportional to $\frac{1}{R}$ on the log-log scale until a saturation region is reached. When the applied force reaches this magnitude, any additional increase in the applied force will not decrease the resistance substantially.



(a) A large force sensitive resistor (b) Resistance vs force chart

Fig.3. A commercially available force sensitive resistor

B *System design with Arduino*

The pressure from the keys pressed was acquired using the *Arduino Leonardo* microcontroller (shown in Fig. 4), which is based on the *ATmega32u4* processor. The *ATmega32u4* is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC (*Reduced Instruction Set Computing*)

architecture that has 32 8-bit general purpose working registers. The AVR is a modified *Harvard* architecture machine where the program and data are stored in separate physical memory systems that appear in different address spaces, but has the ability to read data items from program memory using special instructions

The *Leonardo* has 20 digital input-output pins (of which 7 can be used as PWM outputs and 12 as analogue inputs), a 16 MHz crystal oscillator, a micro USB connection, a power jack, an ICSP header, and a reset button.

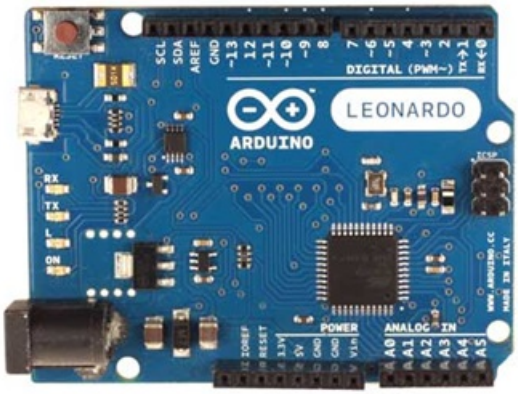


Fig.4. The *Arduino Leonardo*

In order to construct an accurate keystroke biometric pattern, the data acquired was processed by the weightless neural net. The system would capture the force for each key that the user has exerted on the key, and depending on the length of the numeric password, the biometric pattern would have a similar length.

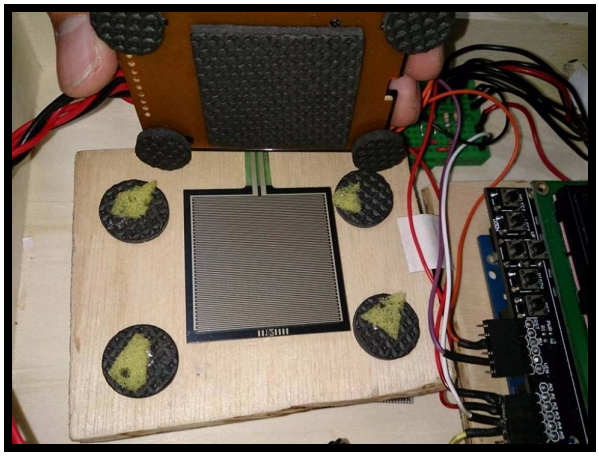
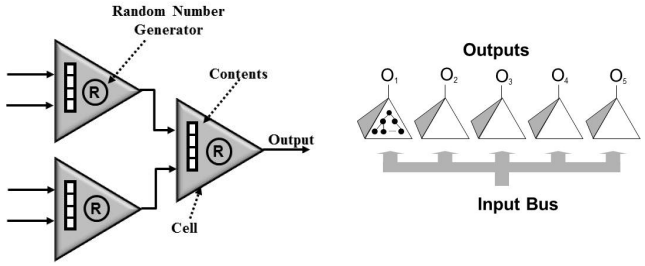


Fig. 5. The system showing the *FSR* in the centre, and the *Arduino Leonardo* on the right

III. WEIGHTLESS NEURAL NETWORKS

Artificial neural networks (ANNs) have been used to perform pattern recognition tasks but this has often been with the multi-layer perceptron network (MLP), trained using the backpropagation learning rule [12]. The weights between the nodes in the various layers are modified by this learning rule. The MLP can deal with nonlinear classification problems because it is able to form complex decision regions (rather than just hyperplanes). Each node in the first layer can create a hyperplane, while the nodes in the second layer combine these hyperplanes to create convex decision regions. In the third layer, the nodes can combine the convex regions to form additional concave regions. Thus, in theory, it is possible to form any arbitrary region with two hidden layers and sufficient hidden units.

However, supervised approaches such as *Backpropagation* are usually considered too slow for many problems where the input dimensionality may be very high and the data sets enormous. Weightless Neural Networks (WNN's) which only require a one pass learning have been around in various configurations since the 1960's. Ludermir *et al* [13][13] give a comprehensive review of the topic. The WNN used here, shown in Fig. 2(a) is derived from the work of Igor Aleksander[14].



(a) The Probabilistic Logic Neuron (PLN)

(b) Typical pyramid structure of the PLN

Fig. **6**. Weightless neural network

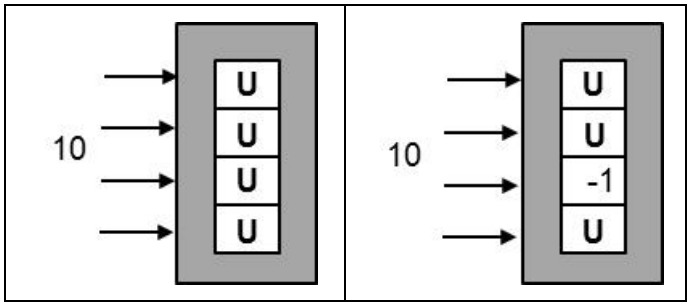
In Fig. 6, we chose a very simple example for the purpose of easy explanation. In reality there may be many input layer cells, with 4, or even 8-bit input address lines. The network will always be triangular shaped (and pyramidal for two-dimensional input data) as the network moves towards the output layer – Fig. 6(b). If we require several output classes, we have to construct the appropriate number of pyramids, each with its own desired output value, but with a common input vector.

During training, the input vector, which (in this case) is a 4 bit binary number, acts as an address generator. Initially, all of the cells' contents in all three cells are set to an undefined state. When an undefined location is addressed, a binary random number generator is called. This produces an output from the cell of either a 1 or a 0 with equal probability. Further developments saw *Filho et. al* [15] proposing a different approach which does not use random number generators. Instead, when an undefined location is selected by the input address to a cell, the output generates an undefined value and this is propagated forward to the next layer. Their *Goal Seeking Neural Network* then provides rules to deal with this.

The idea of having an addressable set and a set of simple rules was developed further by *Bowmaker* and *Coghill* to produce the *Deterministic Adaptive RAM Network* (DARN) [16]. Although the generalisation performance improved, the DARN's capabilities are still poorer than those of the MLP. The Cognitive RAM network (*CogRAM*), which is an enhancement to the DARN has produced very promising results [17]. In the CogRAM, each addressed content is a signed register where all of the registers in the network's cells are initially set to zero. The zero is interpreted as an undefined value. During learning the counter register may become positive, or negative.

A *Learning*

The desired output of every cell in the pyramid is the same as the desired output of the pyramid. If the desired output of the neurons at the input layer is 1, the addressed location is incremented by one. If the desired output is 0, the location is decremented by one. This is illustrated in Fig. 7 where the input pattern '10_2' addresses the location which was initially unassigned (undefined). The desired output of '0' was then stored at this location by decreasing its contents by one - -1.



(a) Before training

(b) Desired output value '0' (or -1) stored in selected location ('*10*')

Fig. **7**. An example of training an input pattern in CogRAM.

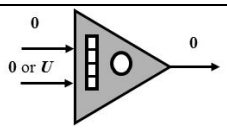
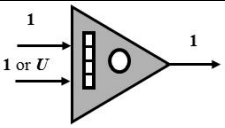
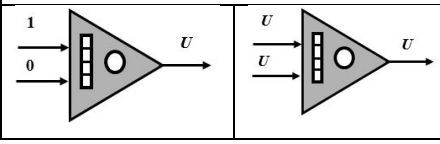
The method then involves calculating the address vectors for the next layer, moving towards the output. The input address vector to each cell in the next layer is constructed by invoking the **Recall** function (described in the next section), on the previous layers from each connecting line in that next layer. The address vector is then applied to the cell inputs of the next layer, and the cell contents are again modified in a similar fashion. This procedure is continued until the pyramid output is reached. The same method is applied to each of the other pyramids, with the entire training pattern set presented just once.

B Recall

In the recall phase, again, the content of each addressed location, starting from the input layer, is interpreted as U (undefined), 0, or 1. In the event that the output of each cell propagates an undefined(U) output forward, the *Goal Seeking Network* (GSN) approach of propagating the addressable set forward to the next layer will be adopted. Even though several locations may be selected at the same time, the following rules, which are essentially the same as for the GSN are adopted.

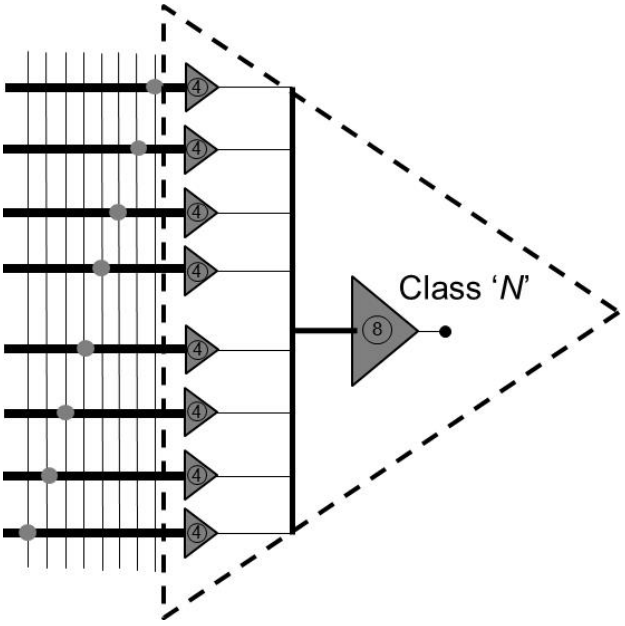
TABLE 1

Recall SCHEME

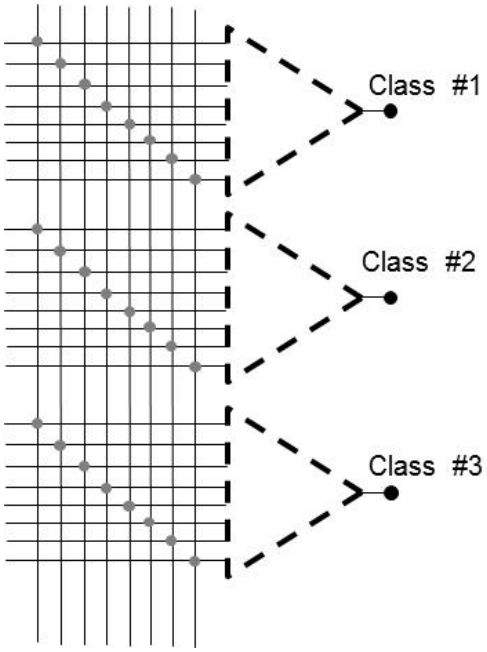
If at least one *zero* is addressed and no *one*s are addressed, the output is *zero*.	
If at least one *one* is addressed and no zeroes are addressed, the output is one.	
Otherwise, the output is *undefined*.	

The Recall operation is propagated forward through the pyramid until the output is reached.

The architecture of the *CogRAM* used for class '*N*' is shown in Fig. 8(a). The input layer consists of cells with 4-*bit* input address lines whereas the next layer has 8-*bit* input address lines. If we have 3 classes to recognize, this basic architecture has to be repeated for total of 3 times. This is illustrated in Fig. 8 (b).



(a) CogRAM for only one class



(b) CogRAM for 3 classes

Fig. **8**. The architecture of the CogRAM used

After the initial training has been completed, there would be many locations in each cell that are UNDEFINED. This is more pronounced in data sets which have a smaller number of training patterns. For the CogRAM, such UNDEFINED cell locations can be reconciled to improve its performance. This is done by looking at each of the UNDEFINED locations in the output cell and computing the nearest address up to a Hamming distance measure of d to a defined location's address (either 0 or 1). This is continued for the whole training set in this way. The Hamming distance d should not be more than half of the cell size. This reconciliation of the UNDEFINED addresses has seen a significant reduction of all the UNDEFINED values in the output cell of each class and *Yee* and *Coghill* [17] have found this contributing to a significant overall improvement in the classification results. However, we believe this may not be a good strategy as two patterns with a Hamming Distance of 1 may not necessarily mean that the two patterns representing the typing force exerted onto the key are similar due to the way in which the input patterns have been coded. A '*slightly*' different force that separates two patterns may end up with a very different binary value. This is illustrated in Fig. 9 where for simplicity of explanation, each of the patterns is represented by a set of 4 features. In (a), the decimal equivalent of the patterns are significantly different even though they differ by just one Hamming Distance. Hence, rather than just changing one bit, we should instead look at the overall picture where the binary representation of the similar pattern should also be reflective of the actual physical value. As an example, a force of 0.7 and 0.9 should share a high degree of similarity if this is the force exerted by the user onto a key. If we use a simple linear transformation scheme where 0.7 is represented as $1\ 1\ 1\ 0_2$ in binary, 0.8 is $0\ 0\ 0\ 1_2$, 0.9 is $1\ 0\ 0\ 1_2$, etc. and we then randomly generate a new pattern of $1\ 1\ 1\ 1_2$. This new pattern has just a one bit difference from the binary coded value of 0.7, and the Hamming distance between the two is 1. This is shown in Fig. 9 (a). Nevertheless, with the simple linear scheme, $1\ 1\ 1\ 1_2$ would represent 1.5. Clearly, the force represented by these two values is not similar, even though their binary patterns may share a high degree of similarity. We believe a more accurate similar pattern is one that would map back to the actual physical representation of the forces in the real world. Thus, a similar pattern would be one that is at the *next force level*(s) which, depending on the binary coding scheme, may not be close in the Hamming distance sense. This is illustrated in Fig. 9 (b).

Pattern	#1	#2	#3	#4
A	1	1	1	0
A'	1	1	1	1

(a)

Pattern	#1	#2	#3	#4
B	1	1	1	0
B'	0	0	0	1

(b)

Fig. **9**. Similar and dissimilar binary patterns

IV. Experimental Setup And Results

In order to validate the *Cognitive RAM* (CogRAM) weightless neural net in testing the keystroke biometrics user authentication system for a numeric keypad, the force sensor was integrated into a microcontroller system so that the force for each key may be captured. The microcontroller acquires the force exerted from each key for the CogRAM to process.

A total of 10 users who were all familiar with using the keyboard and use the keyboard in their daily work were selected. In order to minimise any inconsistent typing rhythm, the volunteers were allowed to choose their own passkey which must be 8 numbers long. The objective here is to see if they would produce a consistent keystroke biometric pattern for the password chosen. This pattern is solely based upon the force exerted on the keys pressed. While shorter passwords are easy to remember, they are not encouraged as they may be easier to hack and gain unauthorised entry too. In addition, the individual patterns may not be unique and inaccurate authentication may result. Longer ones, while providing much better security, can be cumbersome to remember and easy to forget. Each user was requested to key-in their password a total of 6 times. The data was collected in an ordinary everyday environment without any undue stress, thus minimising any inconsistent keystroke patterns. Table 2 (table 2) shows a set of typical keystroke forces for one user.

TABLE 2
An example of a set of keystroke forces

sample	#1	#2	#3	#4	#5	#6	#7	#8
1	*96*	*28*	*50*	*136*	*57*	*98*	*136*	*147*
2	*158*	*32*	*215*	*141*	*22*	*157*	*210*	*189*
3	*187*	*26*	*201*	*180*	*54*	*177*	*56*	*206*
4	*162*	*9*	*12*	*216*	*142*	*136*	*1*	*271*
5	*285*	*76*	*135*	*331*	*70*	*224*	*283*	*335*
6	*96*	*28*	*50*	*136*	*57*	*98*	*136*	*147*

Some pre-processing had to be done so that they may be processed by the CogRAM, - in the range of (0.0, 1.0), using a non-linear sigmoid function (equation (1)).

$$f(x) = 1 \Big/ 1 + \exp(\lambda - x) \qquad (1)$$

where λ, a constant is set to 10. This sigmoid is shown in Fig. 10.

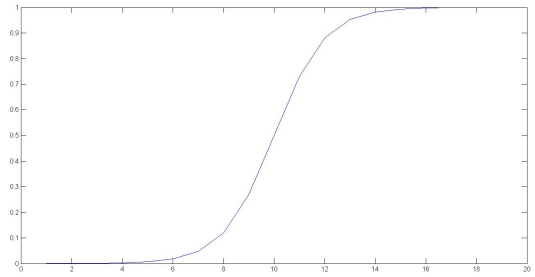


Fig. **10**. A typical sigmoid function

The normalised values were then converted into the binary equivalent form.

One class of pressure patterns at a time was selected for registration, and to train the system. Half of the samples within this class were then randomly chosen to train the CogRAM with the remainder kept for testing.

The accuracy of the system was measured by the *True Positive* (TP) values, - the number of patterns that it was able to identify correctly. In the second experiment, the patterns from the remaining 9 classes were tested against the 1 class with the trained neural net in an attempt to confuse it. This would be used to evaluate the CogRAM's ability to identify such patterns as invalid. This would contribute to the *True Negative* (TN) measure. In a perfect authentication system, these patterns would all be rejected by the system as they are invalid or incorrect - the TN would be 100%. Each class of data was then tested 100 times, each time with a randomly chosen set of training patterns. Fig. 11 shows the average TP and TN values obtained from 100 runs for each pattern class.

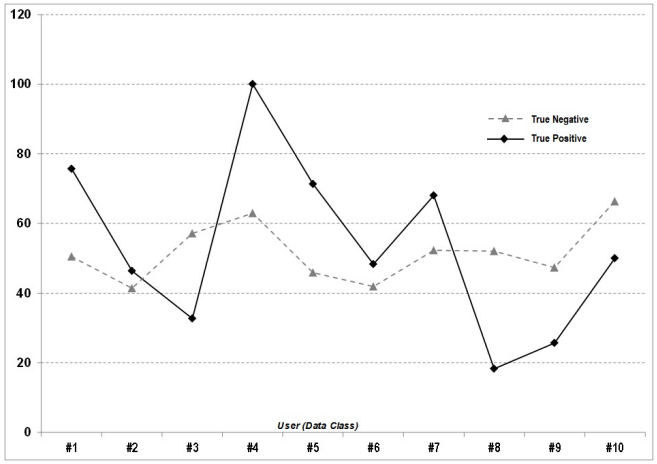
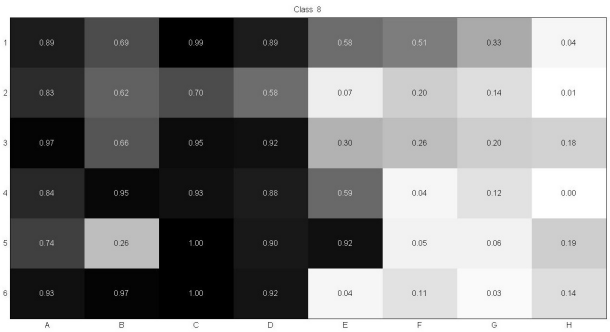
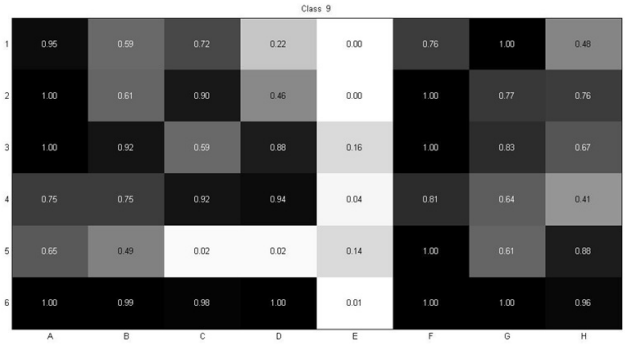


Fig. **11**. Average *True Positive* and *True Negative* results

Most of the *True Positive* (TP) rates are generally acceptable with the exception of those involving #3, #8, #9 and #10 where they are lower than those of the *True Negative* (TN) results. Analysis of the normalized input data graphically shown in Fig. 12 for users #8 and #9 indicates a significant amount of variation in the typing pattern for these users. In Fig. 12, the pressure values for each digit of the password are shown in each column and each of the vertical rows represents a different data sample. Larger values have a darker shade and smaller values, a lighter shade. The maximum value of 1 is represented by black and 0, white. The (normalized) force for each key is shown along the x-axis.



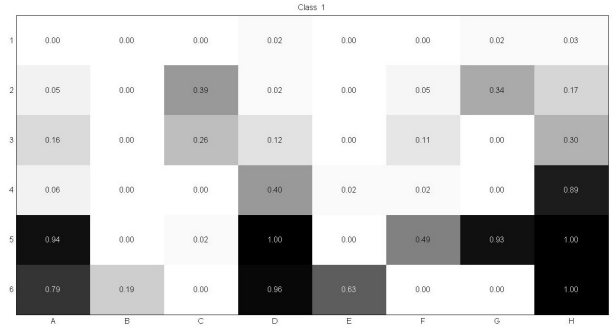
(a) Set #8



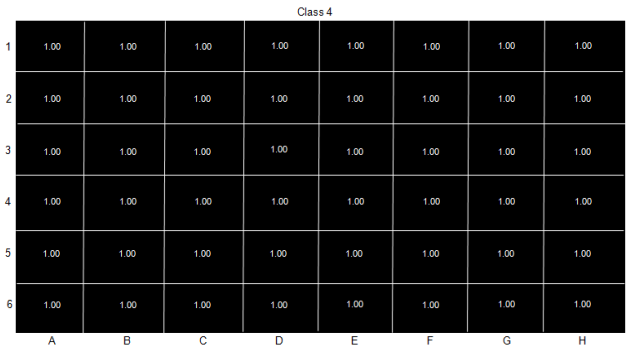
(b) Set #9

Fig. **12**. Irregular typing patterns

However, there are also other users (# 1and #4) which seems to be able to provide a more consistent typing pattern – in the amount of force exerted onto each key. Examples of such users are shown in Fig. 13.



(a) Set #1



(b) Set #4

Fig. **13**. Regular typing patterns

The results of the *False Positives* (FP), - those patterns which have been incorrectly authenticated or accepted, is shown in Fig. 14. These are the results of wrongly assigning the patterns from the remaining users as correctly belonging to the authentic class.

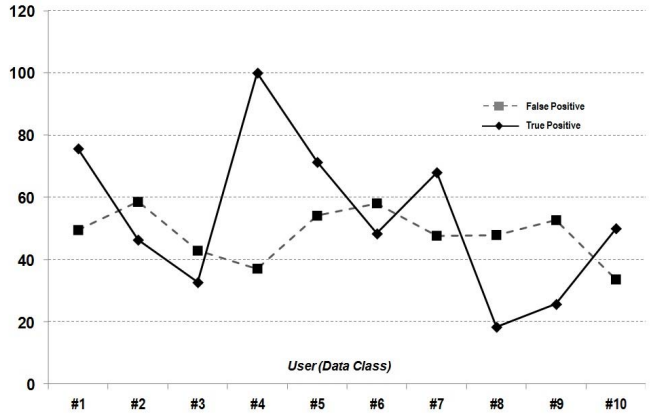


Fig. **14**. Average *True* and *False Positive* results

Finally, we also evaluated the CogRAM's enrolment capability, which is the ability to learn the input patterns. To evaluate this, the same patterns that were used during the learning phase were presented as test patterns on completion of the learning. For perfect enrolment, the system should not have any difficulty in producing the correct output class when presented with these patterns *IF* the patterns have been learnt. The neural net's capability was tested 100 times, each time with a randomly chosen set of training patterns. With the data sets that we have, the CogRAM was able to correctly identify all the training patterns that had been used to train the system and this is shown in Fig. 15.

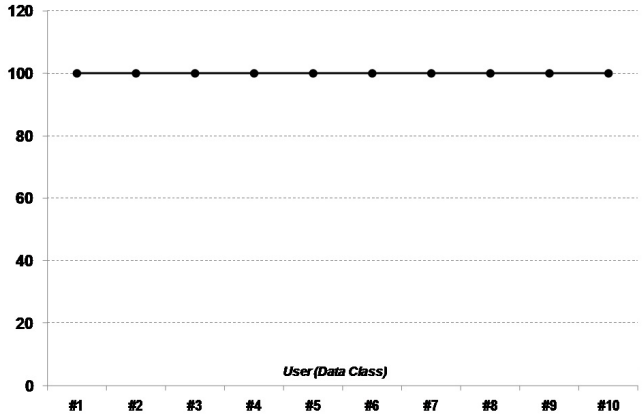


Fig. **15**. Enrollment capability of CogRAM

V. Conclusions And Further Work

Biometrics is needed in modern society because we no longer know every single person in our physical or virtual community – due to either the size of community, or the geographical span of the individuals in the community or a combination of both. Hence, there has been some work done in developing biometrics for authentication or recognition.

Weightless neural networks are known to be weak in

generalization. Hence, to use WNNs for recognition may result in reduced accuracies. However, such systems are able to learn well when trained with a smaller number of different class data. Consequently they are also able to achieve good levels of recognition accuracies. Thus the experiments here were designed with this in mind where the CogRAM was trained with a small set of data and used to authenticate the typing patterns instead. An advantage of using the CogRAM in such an application is that we can easily implement the neural net in hardware and some work has been done in this direction by Nitish and his co-investigators in 2007 [18].

In this paper, we discussed the development and testing of a two-factor user authentication system with the CogRAM weightless neural net. The preliminary results from this piece of work do seem to suggest that the current one-factor password authentication system can be strengthened with an additional factor for numeric keypads. Even though there has been some work being done to investigate the keystroke patterns using either timing or pressure features, these have been conducted with the common QWERTY computer keyboards - where the typing style is significantly different for most people when compared with that used on the numeric keypads. Such keypads as a password entry system do involve a major shift in the typing style. The data captured from the micro-controller system has shown that many of the users were able to generate a repeatable typing pattern based on the force exerted onto each key from the hardware system that was developed. Furthermore, the CogRAM can successfully identify some of the unique and consistent keystroke biometric patterns of the users. Most of the *True Positive* (TP) results are generally acceptable with the exception of those involving #3, #8 , #9 and #10 where they are generally poorer compared with the rest. However, it must be remembered that while biometric technologies can range from the highly physiological at one end, for example, fingerprints, iris, etc., there are also those at the other end of this continuum which exhibit a high level of behavioural traits, for example, signature, keystroke, etc. even though the same behaviour is expected to be repeatable. Hence, the poorer results may be due to the inconsistent typing behaviour of the users in generating the keystroke. The mis-authentication results are also due to the fact that the CogRAM was not able to decide upon the correct class and ended up with the locations that are undefined. Minimising such poor results can be achieved by providing a larger number of typing patterns for the CogRAM to learn. This is an issue that may be investigated further. The most consistent typing pattern was obtained from #4 and the TP results have confirmed this. We have also started preliminary work in developing a standalone hardware implementation of the CogRAM to process the typing data as they are generated in situ. Future work would be directed into investigating how different password lengths may affect the consistency of the user's typing style and resultant pressure patterns for such numeric keypads. The preliminary results from this investigation does seem to indicate that force-based keystroke biometrics for the numeric keypad can provide unique patterns that may be exploited to provide a two-factor user authentication system.

REFERENCES

[1] R. Joyce, and G. Gupta, "Identity authentication based on keystroke latencies," Comm. ACM, vol. 33, pp. 168–176, February 1990.

[2] F. Monrose, and A. Rubin, "Authentication via keystroke dynamics," Proc. of the Fourth ACM Conference on Computer and Communications Security, Zurich, Switzerland, pp. 48–56, 1997.

[3] W. De Ru, and J. Eloff, "Enhanced password authentication through fuzzy logic," IEEE Expert, vol. 12, pp. 38–45, 1997.

[4] M.S. Obaidat, B. Sadoun, "Verification of computer users using keystroke dynamics," IEEE Trans. Syst. Man, and Cybernetics, vol. 27, pp. 261–269, 1997.

[5] E.R. Tee and N. Selvanathan; "*Pin signature verification using wavelet transform*", Malaysian Journal of Computer Science, Vol. 9, No. 2, pp. 71-78, December 1996.

[6] Leenesh Kumar Maisuria, Cheng Soon Ong & Weng Kin Lai, "*A comparison of artificial neural networks and cluster analysis for typing biometrics authentication*", Proceedings of the International Joint Conference on Neural Networks (IJCNN), Washington, DC, July 10-16, 1999.

[7] Chen Change Loy, Chee Peng Lim & Weng Kin Lai, "*Pressure-based typing biometrics user authentication using the fuzzy ARTMAP neural network*", Proceedings of the Twelfth International Conference on Neural Information Processing (ICONIP 2005), pp 647 – 652, Taipei, Taiwan ROC, October 30 – November 2, 2005.

[8] Wasil Elsadig Eltahir, MJE Salami, Ahmad Faris Ismail & Weng Kin Lai 2008, "*Design and evaluation of a pressure-based typing biometric authentication system*", EURASIP Journal on Information Security. Volume 2008 (2008), Article ID 345047, 14 pages, doi:10.1155/2008/345047.

[9] Shereen Yong, W. K. Lai & George G. Coghill, "*Weightless Neural Networks for Typing Biometrics Authentication*", Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information & Engineering Systems, KES'2004, pp. 284 – 293 (Volume II), Wellington, New Zealand, 22nd – 24th September 2004.

[10] M. S. Obaidat and B. Sadoun, '*Keystroke Dynamics Based Authentication*', in *Biometrics: Personal Identification in Networked Society* (Editors: Anil K. Jain, Ruud Bolle and Sharath Pankanti), Chapter 10.

[11] Force Sensing resistor http://en.wikipedia.org/wiki/Force-sensing_resistor accessed on 18th December 2013.

[12] Rumelhart, D.E., Hinton, G.E. and William, R.J., "*Learning internal representation by error propagation"* in Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Volume 1, MIT Press, Cambridge, MA, 1986.

[13] Teresa B. Ludermir et al, "*Weightless Neural Models: A Review of Current and Past Works*", Neural Computing Surveys 2, pp. 41-60, 1999, http://www.icsi.berkeley.edu/~jagota/NCS.

[14] Kan, W.K. & Aleksander, I, "*A Probabilistic Logic Neuron Network for Associative Learning*", Proc. IEEE 1st Int. Conf. on Neural Networks, San Diego, vol. II, pp 541-548, June 1987.

[15] Filho, E.C.D.B.C., Fairhurst, M.C. & Bisset, D.L., "*Adaptive Pattern Recognition using Goal Seeking Neurons*", Pattern Recognition Letters, vol. 12, pp. 131-138, 1991.

[16] Bowmaker, R.G. & Coghill G. G., "*Improved Recognition Capabilities for Goal Seeking Neuron*" Electronics Letters, pp. 220-221, vol. 28, no. 3, Jan 1992.

[17] Paul Yee and George G. Coghill:, "*Weightless Neural Networks: A Comparison Between the Discriminator and the Deterministic Adaptive RAM Network*", KES 2004: 319-328.

[18] Nitish D. Patel, Sing Kiong Nguang, George G. Coghill: "*Neural Network Implementation Using Bit Streams*", IEEE Transactions on Neural Networks 18(5): 1488-1504 (2007).