01 Apr 1991

Identity Authentication Based on Keystroke Latencies Using Neural Networks

Angela Lammers

Sharon Langenfeld

Identity Authentication Based on Keystroke Latencies Using Neural Networks

Angela Lammers, Rockhurst College
and
Sharon Langenfeld, Briar Cliff

Most computer security systems verify the identity of a user through objects in the user's possession such as keys or magnetic cards, or through knowledge the user has, such as a password or PIN number. There are, however, two other methods of user verification, which have as yet received little or no attention. There has been some work done on the third method, recognition of physiological patterns (such as finger prints, retinal patterns, or voice patterns), but this work has been limited and requires expensive hardware to implement. The final method of user verification is through actions such as signature or behavior patterns.

Because each person's signature is unique, it has been used for verification and authentication since man learned to write. This uniqueness is due to the complex physiology of the human hand. The factors that make the human signature characteristic of a single person, also produce a unique pattern of latency times (or time lapsed between keystrokes). It has been shown that this pattern remains fairly constant, especially within words often entered [Joyce & Gupta].

Some work has been done on the use of keystroke latency patterns for user identification and verification. The majority of the systems involve the entry of a particular string several times initially in order to form a mean reference signature of latency times. The standard deviation of this reference signature was used as a measure of tolerance for each login attempt. When a person wants access to the system, he identifies himself by typing his password. The latency vector this produces is compared to the reference signature entered previously, and access is granted if the two are statistically similar [Joyce & Gupta].

Our system makes use of neural networks rather than statistical references in order to classify login attempts as acceptable or non-acceptable. Rather than performing a sequential set of instructions, neural networks are capable of exploring many competing hypotheses in parallel. Because of this quality, neural networks are considered to have the greatest potential in the area of pattern recognition [Lippmann]. The neural network we have developed is trained to recognize a person's unique keystroke latency pattern and will accept or reject subsequent login attempts based on the attempt's approximation of the previously learned pattern.

Inherent in the system is the desire to maximize the number of times the correct user is accepted (called a user success) and the number of times an intruder is rejected (called an intruder failure). At the same time it is desirable to minimize the number of times the correct user is rejected (a false alarm) and the number of times an intruder is allowed access to the system (a break-in).

To make the system more robust, multiple neural networks are trained and tested. One neural network is located at each processor, thus parallelizing the learning and testing. Each processor or neural network receives its own unique initial weight vector which is produced by a random number generator. The neural networks are trained on the inputs and each ends up with its own unique learned weight vector depicting the user's signature.

The first step involves educating the neural networks by entering a password string many times and teaching each neural network to recognize the string. As the string is typed in repeatedly, the system records the time between keystrokes. These times are used to train the neural networks, one located at each processor of the system, to recognize the latency pattern.

Neural networks learn the electronic signature of the user by adapting the set of weights associated with the input until the result is within a desired range [fig. 1]. The initial weights are obtained through a random number generator. After the learning has taken place, these weights represent the digital signature of the user.

Each neural network learns by taking the inputs (the array of latency times) and multiplying each of the elements of the time vector by its associated weight vector and summing:

$$R=\Sigma X_i W_i$$

where R is the result, X the input vector, and W is the weight associated with the input [fig. 2].

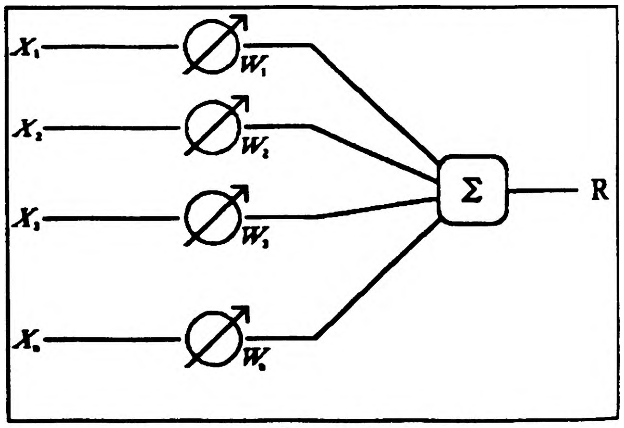


Figure 1

If the result of the summation converges to 1 (the desired output), then the neural network is said to have learned the inputs. The network converges due to adjusting the weights of the neural net and sending the inputs back through the network again.

The weights are adjusted accordingly: The error ε is calculated by subtracting the result obtained (R) from 1 (the desired result):

$$\varepsilon_i=1-R$$

The change in weight W is then found using the equation:

$$\Delta W_i=a\varepsilon(X_i-\varepsilon W_i)$$

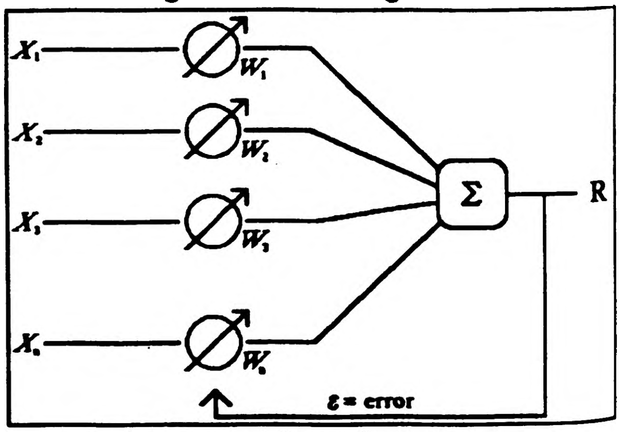


Figure 2

This function is a variation of the Hebbian rule:

Our function has some interesting properties including:

1. It looks like a Hebbian algorithm with error instead of output.

$$\Delta W_i = a \varepsilon X_i$$

2. It reduces weights and keeps them from growing too large.
3. It controls excessive jumps in response to the error.

We found our function caused the output to converge to the desired output (1) rather quickly.

Once the networks have learned the user's signature, the program requests attempts for access to the system. Each attempt is taken as input to each of the trained neural networks. The inputs are multiplied by the learned weights and summed to produce a result. If the result of an individual network falls within the allowed acceptance range, the processor sends a message of acceptance back to the controller. If the majority of the processors accept the signature, the user is granted access to the system and the attempt is integrated into all the networks. If there is not a majority acceptance, the user is denied access and the attempt is discarded.

In the experimentation with the program, many parameters were varied. First, the size of the acceptance range and the size of the password were tested. Not surprisingly, using small passwords (3 characters in length) and a small acceptance range (.90 to 1.10) rendered the most break-ins and the most false alarms. Meanwhile, using medium sized passwords (7 characters in length) and a bigger acceptance range (.85 to 1.15) yielded the same number of break-ins, but fewer false alarms. Using big passwords (9 characters in length) and a small acceptance range (.90 to 1.10) produced no break-ins, but incurred the greatest number of false alarms. The best alternative was a medium sized password (7 characters in length) and a small acceptance range (.90 to 1.10) This combination provided few break-ins and even fewer false alarms.

Next, the use of input to the system was altered. Originally, the neural networks were trained on five separate inputs from the user -- one after the other. This, however, invoked the problem of an inaccurate digital signature if the last password entered was not indicative of the user because of a pause or slip during entry. Because the last entry was learned most recently, the neural network looked for the inaccurate pattern during the attempt stage. Therefore, the average of the inputs was used as a basis for training the neural networks. Not surprisingly, the average of the inputs produced more user successes than did the sequential learning of the individual passwords, while maintaining the same number of break-ins.

Finally, the function in the learning algorithm was altered. Using a linear function:

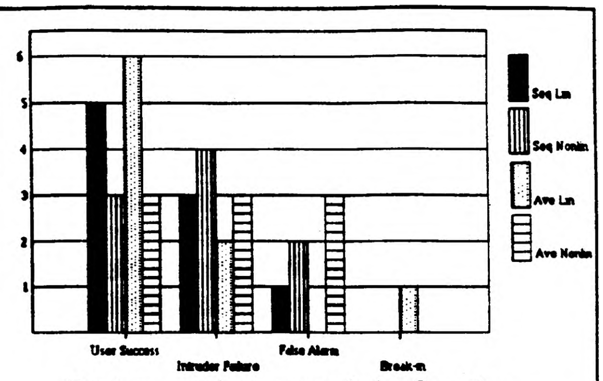
$$R = \Sigma X_i W_i$$

where R is the result, X is the input vector, and W is the weight vector, provided 10% more user successes than did the nonlinear function:

$$R = \Sigma X_i^2 W_i$$

where the input vector was squared before being normalized. Altering the function type had no effect upon the number of break-ins incurred: as user successes increased, false alarms decreased.

Therefore, the best combination of the parameters tested are: taking the average of the inputs of medium sized passwords trained on neural networks containing a linear function and tested in a small acceptance range. Figure 3 shows the number of user successes, intruder failures, false alarms, and break-ins for each of the four program types we tested (sequential inputs with a linear function, sequential inputs with a nonlinear function, averaged inputs with a linear function, and averaged inputs with a nonlinear function). Tests shown were conducted with mid-sized password and small acceptance range. Our tests overall produced a 75% success rate (user successes and intruder failures). Of the 25% failure rate, only 3% were due to break-ins. (22% were due to false alarms.)



System result-type totals by function

Figure 3

From our results we are able to conclude that keystroke latency is a valid measure of security when implemented with password checking and other common security measures. Neural networks not only make the implementation of the keystroke latency security system easier, but also grow and adapt with each user--thus making them superior to statistical keystroke security systems by increasing the flexibility of the system. Parallelizing the system makes it more robust than when implemented on a single processor since more than one neural net is trained and tested on each input.

REFERENCES

Joyce, Rick and Gopal Gupta, "Identity Authentication Based on Keystroke Latencies", Communications of the ACM 33 (2) (Feb '90) 168-176.

Lippmann, Richard P., "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine (Apr '87) 4-22.