Application of MultiLayer Perceptron Network as a One-Way Hash Function

Liew Pol Yee[1] and Liyanage C. De Silva[2]

National University of Singapore, Department of Electrical and Computer Engineering
4 Engineering Drive 3, Singapore 117576
[1] cellomuse@yahoo.com, [2] elelcds@nus.edu.sg

***Abstract* –In this paper, the applicability of using a MultiLayer-Perceptron (MLP) Network as a possible hash algorithm is investigated. The difficulty of recovering an input from an MLP Network hashed output is presented. Important features of good hash algorithms such as resistance to birthday attacks and collision free hashing are explored with regard to the MLP Network. Possible advantages of using such an arrangement over existing hash algorithms are mentioned.**

***Keywords* – Cryptography, Neural Network, One Way Hashing.**

I. INTRODUCTION

A one-way hash function is an important element of cryptography. Also called a compression function, contraction function, message digest, fingerprint, cryptographic checksum [1], it provides important functions such as the authentication of files and passwords. A hash function takes in a variable number of bits as input, and produces a fixed number of bits, usually 128, as output. A good hash function makes it impossible to recover the original input from the output. A good hash function also makes it difficult to find another set of input bits other than the original that will produce the same output. Thus, authenticating a password simply requires comparing the hash of the password with a stored copy of the same hash. Yet, a compromised system containing the stored hashes will not be able to reveal the original passwords.

Available hash functions include the Message Digest family comprising of MD2 [2], MD4 [3] and MD5 [4] by Ron Rivest, and Secure Hash Algorithm (SHA) [5] by NIST and NSA. Other functions are available, though MD5 and SHA are more commonly used. While both MD5 and SHA can be considered improved versions of MD4 with better avalanche effects, SHA differs from MD5 by hashing to 160 bits instead of 128 bits.

In this paper, an investigation is made on the applicability of using an MLP Network as a one-way hash function. In particular, the MLP Network will be analyzed regarding its ability to satisfy the conditions of a good one-way hash function: difficulty of obtaining the input from the output, resistance to birthday attacks and collision resistance.

In part 2 of the paper, there will be some preliminary introduction on neurons and sigmoidal activation functions. The structure of the MLP Network used for the one-way hashing function will be developed and presented. This is followed by an analysis of using the MLP Network as a hash function.

It is not encouraged that the one-way hash function presented be used today in any critical or sensitive data or application. The strength of any new hash function can only become apparent after intense public scrutiny.

II. PRELIMINARIES

In this section, an overview of the concepts behind neural networks and MLP Networks in particular will be presented. Artificial neural networks (ANN) are mathematical models of complicated biological neural networks. ANN consists of many computation components in parallel. ANN can also be trained, and together with the potential to process vast amount of data in parallel, can be used for many purposes including pattern recognition and economic prediction [6].

A. Neural Networks

Figure 1 presents the structure of a neuron [7] that will be used to build the MLP Network.

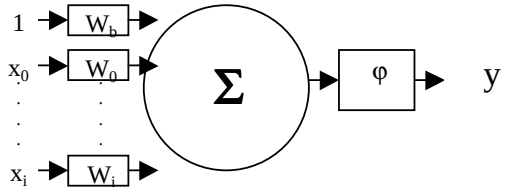

Figure 1: Structure of a Neuron

The output of a neuron is given by the following equation:

$$y = \varphi[(\sum x_i \times W_i) + W_b] \quad (1)$$

where x_i are the input bits, W_i are the corresponding weights of the neuron, W_b is the bias weight and φ is the activation function.

The activation function is a non-linear sigmoidal function.

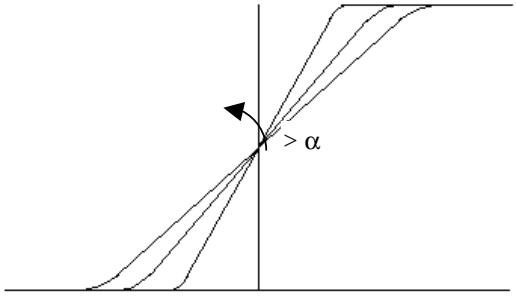

Figure 2: Sigmoidal functions with increasing α

The sigmoidal function is given by the following equation:

$$\varphi(v) = \frac{1}{1+e^{-\alpha v}} \qquad (2)$$

where α determines the slope of the sigmoidal function.

The sigmoidal function replaces the step function found in the perceptron of McCulloch and Pitts. It is thought that biological neurons generate a continuous action potential in response to a particular level of depolarizing current X [8]. A continuous sigmoidal function thus makes a better representation rather than a step function.

Non-linear functions are used in most if not all one-way hash function. Non-linear functions make it difficult to obtain the input from the output function. In MD5, there are a total of four non-linear functions that takes in three 32-bit inputs and produces a 32-bit output. The presence of the sigmoidal non-linear function suggests that the MLP Network may be suitable for one-way hashing. This will be further explored during the analysis of the MLP Network as a one-way hash.

B. Structure of MLP Network for one-way hashing

The structure of an MLP Network is made up of layers of neurons [9]. For the purpose of the one-way hashing function, two layers of neurons are employed, the preceding layer is called the "hidden" layer, and the other called the "output" layer. The meaning of the words "layer", "hidden", "output" and "input" differ in terminology for different text. For our purpose, the term "input layer" is not actually a layer of neurons, but rather the input bits themselves. The graphical structure of a fully connected MLP Network is shown in figure 3.

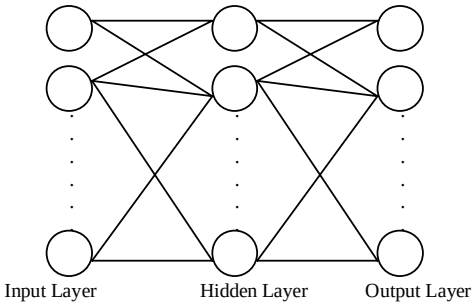

Figure 3: Structure of an MLP Network

Input bits are fed into the "hidden" layer of neurons. The output of the "hidden" layer becomes the input for the "output" layer. Due to the sigmoidal function, the output of each neuron is a real number between 0 and 1. Since it will be more useful to have binary values of either 0 or 1, a threshold of 0.5 is taken. Real outputs less than 0.5 will be taken as 0, while those greater or equal to 0.5 will be taken as 1.

The weights W_i are taken from a consistent source. A good source of weight values can be generated from pseudo-random number generators that follow a gaussian distribution with a mean of 0 and a variance of 1. It is generally advisable to use values that are between –1 and 1 to prevent a neuron from being saturated. Another possible source of weight values may be obtained from random tables, though the values should again be normalized so that they lie between –1 and 1. The values of the weights in the MLP Network is required to be the same every time the MLP Network is initialized for use as a one-way hash function.

The model of the MLP Network used in this paper has an output layer consisting of 128 nodes, with a hidden layer consisting of 64 nodes. The number of input bits is set to 640. The total number of weight values including bias weight values, n, required is given in the equation below.

$$\begin{aligned} n &= (128\times 65) + (64\times 641) \\ &= 49344 \end{aligned} \qquad (3)$$

The MLP Network will roughly require between 1.5 to 3 megabytes of memory in order to store the weight values. This may not be feasible in smart card systems, though in software, it will not be a problem. It should also be possible to develop a hardware chip that performs the functions of the MLP Network with built in memory space for weight storage. Existing hardware includes Accurate Automation Corp's Neural Network Processor and Intel's 80170NX Electrically Trainable Analog Neural Network (ETANN).

C. One-way hash operation

Since the input can be of variable number of bits long, padding is done in a similar fashion performed in MD5. A single '1' bit is appended followed by '0' bits until the length of the message is congruent to 448 bits. This is followed by a 64-bit representation of the original message length. This technique is known as MD-strengthening [10], and overcomes security problems arising from messages of different lengths hashing to the same output. Padding is done also because 512 bits of data are manipulated at a time during the one-way hash process.

The hashing process proceeds by taking the 512 bits of data at a time, combine the input with the 128 bit output of the previous round, and obtain the 128 bit representation. The process stops when no more 512-bit blocks of data are available. The last 128-bit output is the hash representation of the input. A known initialization vector is used at the start of the process.

III. ANALYSIS OF THE MLP NETWORK AS A ONE-WAY HASH

In this section, the three important aspects of one-way hash are explored in relation to the MLP Network.

A. Difficulty in recovering input from the hashed output

Essentially, a one-way hash function is thus named because it only allows a representation to be created from the original message, yet the representation cannot be used to recover even a single bit of the original message, also known as Pre-image resistance [12]. This is important as one-way hash functions are used in digital signatures. Instead of signing a long message, which can take a long time, the hash of the long message is signed instead.

The MLP Network can be represented in matrix form as shown below.

$$\begin{bmatrix} W_{1,b} & W_{1,0} & \cdots & W_{1,639} \\ W_{2,b} & W_{2,1} & \cdots & W_{2,639} \\ \vdots & & \ddots & \\ W_{64,b} & W_{64,1} & \cdots & W_{64,639} \end{bmatrix} \times \begin{bmatrix} 1 \\ i_0 \\ \vdots \\ i_{639} \end{bmatrix} = \begin{bmatrix} o_0 \\ o_1 \\ \vdots \\ o_{63} \end{bmatrix} \quad (4)$$

$$\begin{bmatrix} W_{1,b} & W_{1,0} & \cdots & W_{1,63} \\ W_{2,b} & W_{2,1} & \cdots & W_{2,63} \\ \vdots & & \ddots & \\ W_{128,b} & W_{128,1} & \cdots & W_{128,63} \end{bmatrix} \times \begin{bmatrix} 1 \\ o_0 \\ \vdots \\ o_{639} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{127} \end{bmatrix} \quad (5)$$

Where y_0 to y_{127} is the result of the output layer, o_0 to o_{63} is the result from the hidden layer and $W_{x,y}$ is the weight corresponding to neuron x in the layer and input y.

Take **i, o,** and **y** to be the array of input bits, hidden layer output bits and output layer output bits respectively. From equations 4 and 5, it seems possible to find **o** from **y** simply by performing Gauss-Jordan reduction [11]. This can be followed by performing Gauss-Jordan reduction again to find the input **i**.

Luckily, the sigmoidal function makes finding the unknown **o** and **I** extremely difficult. Using the knowledge that the output elements of **o** lie between the range of 0 and 1 since it is the output of a previous sigmoidal function, and by increasing the value of α in equation 2, it is possible to modify the sigmoidal function such that the probability of finding the inverse be extremely small. As α increases, the sigmoidal function approaches that of a step function shown below.

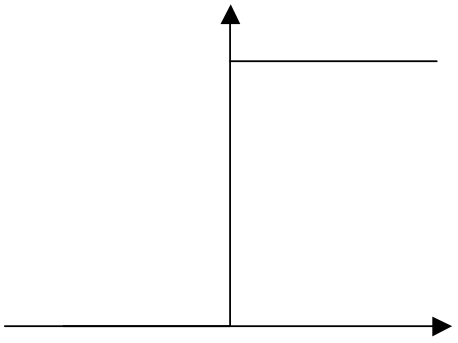

Figure 3: Step function

Supposing that a probability of 5×10^{-2} percent be required for the output to be 0.9999999999999999999, the value of α is found by:

$$\begin{aligned} \alpha &= -20\ln\left(\frac{1}{0.99999999999999999999} - 1\right) \\ &\approx 921 \\ &= 1000 \end{aligned} \quad (4)$$

By setting α to be 1000, it is possible to reduce the probability that an output is not equal to 1 or 0 to 0.1%. Extending this theory, the probability that none of the bits is equal to 1 or 0 will be 0.001^{128}, which is a very small probability. Due to this small probability, Gauss-Jordan reduction will no longer work should α be set to 1000 for the output layer. The value of α for the hidden layer can be set to a much smaller number since it is not necessary to drive a neuron into saturation.

B. Collision resistance and Resistance to Birthday attacks

Collision resistance is the difficulty of finding a different input that will map to the same output after hashing, for example H (M') = H (M). As the hash is supposed to provide a "unique" representation of the input data, allowing another disparate input data to hash to the same output defeats the purpose. Birthday attacks are similar in idea, except that instead of finding another input that will hash to a required output, two random input data are found such that they hash to the same output. Birthday attack resistance is also known as 2nd Pre-image Resistance [12]. Thus it should require a search though 2^{64} before two such inputs that hash to the same output are found.

In section 3.1, it was determined that α for the output layer of neurons be set to 1000 to increase the difficulty of finding **o** from **y**. However, this also means that multiple **o** may be able to map to the same **y**, for example, **o**[0] may be 0.12345678 or 0.12345679 and yet be able to produce the same **y**. This possibility suggests that the MLP Network be susceptible to collision.

In order to prevent this from happening, it must be made such that a single bit change in **i** will result in a vast change in all the bits of **o** such that the previous example cannot occur. Each neuron in a layer within a fully connected MLP Network is totally connected to all the neurons in the next layer. This inherent structure expedites the avalanche effect, as a change in a single bit of the input will definitely result in changes in the output of a neuron layer. Using the fact that **i** can only consist of 0 or 1, the value of α for the hidden layer neurons is set to 1, while the weights of the hidden layer neurons are truncated to 3 decimal places. This guarantees that any change in **i** will result in **o** at 4 decimal places.

C. Converting the One-way hash into a MAC

A Message Authentification Code (MAC) is a one-way hash with an additional secret key. The MLP Network used as a one-way hash function provides two ways in which it can be converted into a MAC. The first, similar to other hash functions, is through the initialization vector used at the beginning of the hashing operation. The initialization vector becomes a 128 bit key that can be used to turn the one-way hash into a MAC. The second way is by providing an input key and a training output key. Using these keys, back-propagation algorithm is applied to the MLP Network to train and adjust the weights of the MLP Network. The MLP Network becomes unique to the input and output key.

D. Advantages of Proposed Application

Applying an MLP Network as a hash algorithm presents several useful attributes. Firstly, simply changing the initial weight values of the MLP Network employed results in a totally different hash output. Many hash algorithms can be implemented for different purposes by setting unique initial weight values for each purpose. Secondly, the MLP structure can be modified, allowing for a hashed key of more than 128 bits, simply by adding more neurons into each layer.

IV. CONCLUSIONS

In this paper, the idea of using an MLP Network as a one-way hash function is explored. The MLP Network structure developed for one-way hashing consists of a hidden layer and an output layer. The hidden layer contains 64 neurons with 641 inputs including the bias. The weights of the hidden neurons are truncated to 3 decimal places, while α is set to 1. The output layer contains 128 neurons with 65 inputs including the bias, with α set to 1000. The MLP Network thus structured is shown to be pre-image resistant, 2^{nd} pre-image resistant and collision resistant. The proposal of applying the MLP Network as a hashing algorithm has several advantages over existing algorithms, as it can easily be adjusted to produce variable number of output bits, and it can also be initialized uniquely for multiple purposes.

References

[1] B. Schneier, "Applied Cryptography", Chap 2, pp 30-31, John Wiley, ISBN 0-471-11709-9, 1996.

[2] R. Rivest, "The MD2 Message Digest Algorithm", RFC 1319, MIT Laboratory for Computer Science and RSA Security Inc., April 1992.

[3] R. Rivest, "The MD4 Message Digest Algorithm", RFC 1320, MIT Laboratory for Computer Science and RSA Security Inc., April 1992.

[4] R. Rivest, "The MD5 Message Digest Algorithm", RFC 1321, MIT Laboratory for Computer Science and RSA Security Inc., April 1992.

[5] National Institute of Standards, and Technology, NIST FIPS PUB 186, "Digital Signature Standard", U.S. Department of Commerce, May 1994.

[6] W. S. Sarle, "Neural Network Faq", ftp://ftp.sas.com/pub/neural/FAQ.html, 2001.

[7] W. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", Bulletin of Mathematical Biophysics, Chap 7, pp 115-133, 1943.

[8] P. Crochat and D. Franklin, "Back-Propogation Neural Network Tutorial", http://ieee.uow.edu.au/~daniel/software/libneural/BPN_tutorial/BPN_English/BPN_English/, 2001.

[9] S. Haykin, "Neural Networks: A Comprehensive Foundation", Chap 4,pp 178-184, Prentice Hall, ISBN 0-13-273350-1, 1999.

[10] B. Schneier, "Applied Cryptography", Chap 18, pp 431-432, John Wiley, ISBN 0-471-11709-9, 1996.

[11] P. V. Oneil, "Advanced Engineering Methematics", Chap 8, pp 355-371, International Thomson Publishing, ISBN 0-534-94320-9, 1995.

[12] A. J. Menezes et al, "Handbook of Applied Cryptography", Chap 9, pp 323-324, CRC Press, ISBN 0-8493-8523-7, 2001.

[13] Michael Roe, "Performance of Block Ciphers and Hash Functions – One Year Later", Fast Software Encryption 94, pp359-362, 1994.