Image Hash using Neural Networks

Veena Desai
Research Centre ,Dept of E&C,
GIT, Belgaum, India

D H Rao, PhD.
Jain College of Engg,
Belgaum, India

ABSTRACT
Hash functions have been used to generate hash codes for data authentication. Traditionally these functions are generated using byte oriented algorithms like MD5 and others. In our paper we propose a new method of generating hash code for images using neural networks. Three sample images namely, fingerprint, lena and football image have been considered and their hash values calculated using two neural network structures namely, 1) structure without feedback 2) structure with feedback. The original images are then subjected to bit modification, Gaussian noise and rotational noise. The hash values are recalculated for the modified images. Sensitivity and hit collision are calculated and are found to be comparable with that of MD5 algorithm.

General Terms
Neural Networks, Cryptography, Security

Keywords
Image hash, neural hash, hash sensitivity, hit collision

1. INTRODUCTION
A hash function H is a transformation that takes a variable-size input message M and returns a fixed-size hash string H, which is called the hash value. A Hash function H(M) generates a unique hash value H for a particular image and thus can be used for checking data integrity and authentication purposes. When image messages are considered preimage resistance and hit collision are parameters for evaluating the performance of hash functions.

The applications of neural networks in areas of cryptography in general are discussed in [1-8].The authors of [9-10] have used both chaos and neural networks in data encryption because of their cipher-suitable properties, such as parameter-sensitivity, time-varying, random-similarity, etc. Based on chaotic neural networks, a hash function is constructed, which makes use of neural networks' diffusion property and chaos' confusion property. This function encodes the plaintext of arbitrary length into the hash value of fixed length (typically, 128-bit, 256-bit or 512-bit).

Another demonstration of hash function implementation based on conservative chaotic system is proposed by authors of [11]. In their implementation the plaintext is divided into a group of message blocks by a fixed length and each message block is iterated some times through standard map. Both the iterations results of every round and the plaintext block determine the two initial values and the steps of iterations in next round. Some items of the result in the final round are chosen to be transformed into hash value of 128 bits. In the paper [12] a hash function construction method based on cellular neural network (CNN) with hyper-chaos characteristics is proposed. The chaos sequence generated by iterating CNN with Runge-Kutta algorithm, then the sequence iterates with every bit of the plaintext continually. Then hash code is obtained through the corresponding transform of the latter chaos sequence from iteration. Hash code with different lengths could be generated from the former hash result. In [13] 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 61 input including the bias. The weights of the hidden neurons are truncated to 3 decimal places, which α set to 1000. The MLP network thus structured is shown to be pre image resistance, 2nd pre image resistance and collision resistance features.

In section 2 of our paper the neural network structures used in the proposed implementation are illustrated and explained. Section 3 provides the proposed algorithm details. Result calculation and sample data are elucidated in section 4. Conclusions are discussed in section 5.

2. NEURAL NETWORKS
Artificial neural network is an interconnected group of artificial neurons which use a mathematical model or a computational model for information processing based on connectionist approach to computation. It is a network of simple processing elements which can exhibit complex behavior determined by the connections between processing elements and element parameters. Artificial neural network is an adaptive system that changes its structure based on external or internal information that flows through the network.

The method of setting the values for the weights enables the process of learning or training. The process of modifying the weights of the connections between network layers with the expected output is called training a network. The internal process that takes place when a network is trained is called learning. Figure 1 and Fig.2 show the neural network structure implemented in this paper without feedback and with feedback respectively.

The structure of feed-forward network as in Fig. 1 is made up of layers of neurons. For the purpose of the one-way hashing function, three layers of neurons are employed. The first layer is called ‘input’ layer, the next is the ‘hidden’ layer and the last layer is the ‘output’ layer. In our implementation, the input and the hidden layers consist of 128 neurons and the output layer consists of 64 neurons.

A sequence of binary bits are obtained from the image as mentioned in section 3. These bits are elements of out(1,n) from the input layer and are processed further by the hidden layer. The output of hidden layer becomes the input for the output layer. Due to the use of tan-sigmoidal function, non-linearity is introduced in each neuron of the output layer. In our structure, we have used a constant weight of 0.5 for each neuron and the bias is set to 0 for each neuron. Diffusion property is also satisfied due to the use of neural structure.

The structure of feedback network is again a structure of 3 layers of neurons. The interconnect between input and hidden layer output are as in feed forward structure.
Fig 1. Neural network structure without feedback

Fig 2. Neural network structure with feedback
Each output neuron is mapped from the output layer to two consecutive input layer neurons. For each iteration the input layer neuron takes into consideration the current input and previous output due to which stability is introduced in the structure thereby making it more immune to noise.

3. PROPOSED ALGORITHM

For neural network structure without feedback

1. Read the input image
2. Convert the image into two dimensional pixel values
3. Determine the number of rows and columns of image
4. Create a row matrix of content values
5. Resize the row matrix into a matrix of size(r * 128). (r is the number of rows and it depends on the image which is being read)
6. Initialise each neuron in the input layer, hidden layer and output layer with a constant weight of 0.5 and a bias of 0.
7. Initialise the final layer output \( y_1(1), y_1(2), \ldots, y_1(64) \) to zero
8. For k = 1 to r
9. \( \text{If}(k=1) \)
10. Let the final layer output be \( y_1(1), y_1(2), \ldots, y_1(64) \)
11. End if
12. \( \text{If}(k>1) \)
13. \( b_{k-1} = [b_{k-1}(1) \text{ EXOR } y_1(1)], [b_{k-2}(2) \text{ EXOR } y_1(2)], \ldots, [b_{k-2}(64) \text{ EXOR } y_1(64)] \)
14. End if
15. End for k
16. The hash value of image without noise is \( h_{k-1} \)
17. Modify the image by introducing one of the following four noises: One-bit change / Gaussian filtering / +5 degrees rotation / -5 degrees rotation.
18. Repeat Steps 7 to 15 for the modified image.
19. Let the hash of the modified image be \( h_{k-1} \).
20. Calculate the difference between \( b_{k-1} \) and \( h_{k-1} \).

For neural network structure with feedback

Repeat Steps 1 to 7 of without feedback

1. For k=1 to r
2. \( \text{If}(k>1) \)
3. Initialise t to 1
4. for j = 1 to 128 in steps of 2
5. \( \text{out1}(k,j) = \text{EXOR}[y(k-1,t)\text{out1}(k,j)]; \)
6. \( \text{out1}(k,j+1) = \text{EXOR}[y(k-1,t)\text{out1}(k,j+1)]; \)
7. \( t=t+1; \)
8. \( \text{if}(t == 65) \)
9. End for j
10. End if
11. End for k
12. Repeat Steps 16 to 20 of Without Feedback

The input image is converted into two dimensional pixel values. 128 values of this matrix are given at a time to the input layer of the neural network. These values are passed through the hidden layer and 64 values of 38 bits each are obtained from each neuron of the output layer.

For the without feedback neural network structure, the output obtained from the consecutive iterations are XOR’ed to get the final hash value for the particular image.

For the with feedback neural network structure, the values generated by each output neuron for the previous iteration are XOR’ed with the input values to two consecutive neurons in the current iteration. Let the hash value generated for the original image for both without feedback and with feedback structure be \( h_1 \). Noise is introduced into the image and the procedure is repeated to obtain the new hash for the modified image. Let this hash be represented as \( h_2 \).

Sensitivity gives the number of bits change in the original image after addition of noise and is calculated as:

\[
%\text{Sensitivity} = \frac{(\text{Number of bits changed})}{(\text{Total number of bits})} \times 100
\]

Hit collision is the number of digits (hex values) remaining same in the hash after addition of noise in the original image and is calculated as:

\[
%\text{Hit collision} = \frac{(\text{No. of hex values remaining same})}{(\text{Total No. of hex values})} \times 100
\]

4. SAMPLE DATA AND RESULTS

The following figures, 3(a) to 3(e) show the sample set of images that have been used for hash calculation. Fingerprint image 1(a) is the original image of sample 1. Fig. 4(b) to 4(e) are the modifications on sample 1 with 1 bit change, filtered image with +5 degree rotation, and -5 degrees rotation respectively.
Hash value of fingerprint image:

BCC85FBC44BB32600A960A6EB3C1B538590D2254B4
8A4261E1290D73A67C360363684A938C2C8E7E687400CB
7FB2482D0C3BA6EB56BD379FB766FF3EEE31216A0541
C9AFBCA72C0A0F8F8EB48752A465C01B70BB8EC74
B7CDAD8127B5E41DA38BA97BCD1DFA21A693507C6
CAEBFBE49BA9D26FF2A24B7615B37A99F738E0A6F6CF6
7038559AE920A4D5F5E76E84D20C836A35FDF89845C3
7E648D7735EB11B62A18937C395FAE9BCEO1D810A
B73S57321538CA765C81E55C0B02B723E2CE5E7B57108
9A6AD08DEC6F5D109831F39F74FD65232153DEEEF6
4C2CD9A2FC31A38482513EECFC02746B0F2A6616B
A99ABD4211579786CF06ECC4EB08841CC1425A6BA3239
9518FS3AE07C69021243FFE5A835A989AD18963B21BA
9E47AECFC28AF19E7E40E0A0F400077B

Hash value for fingerprint image with 1-bit change without feedback structure:

130F87837BCE576E8E0298AE7D1348905DE58003DF2EFC
B0D8E41F2A72A4B880B793176DA3E8051D508B181EE692
5A3A83BFB45D42D688353C7EEBA90A8F30DFE32343D94AD
187D9BE02308AB318A0CA136AE811FF8301714FEEC8522
D516DE644997738C111A6F646858E20880ED2A6F55AE89
0A0CC0DAEE3785247CC83E621A928FDBAEE4852EA482E00
0000024AB942A01E927101FFBC733A65EE3B3C832AAOE
EC3712924FDCBC8CB4F4E55E03A3E69090F4A69A1D9DE26
C40C514052EEFA9D2F729A53409FE5F81BD1EAD829CC8FBA
9101FA2C8F16147974076130A417DEBAE41BD80A0ED92A59
654EC88ADA772259081BE849F4AC0DDBE82882919D480CF
F496A363BB07C94855DEB74A6267A5683B666FEF38FDA
EC48BAC51EAB2424D65EAE3C05521C7

Number of bit changes observed in the New hash: 1115
Sensitivity: 54.8470 %
Hit Collision: 0.10329 %
Elapsed Time: 58.32 minutes

Table 1 provides the comparison of sensitivities for hash values obtained from the neural network without feedback and MD5 algorithm for the fingerprint image and Lena image. In both cases we find that the sensitivity of the hash obtained from the neural network is comparable with that of MD5 algorithm.

<table>
<thead>
<tr>
<th>Type of noise</th>
<th>Sensitivity (MD5 Tool)</th>
<th>Sensitivity</th>
<th>Sensitivity (MD5 Tool)</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-bit change</td>
<td>57.03 %</td>
<td>48.27 %</td>
<td>57.03 %</td>
</tr>
<tr>
<td>+5 degrees rotation</td>
<td>47.65 %</td>
<td>49.79 %</td>
<td>49.79 %</td>
</tr>
<tr>
<td>-5 degrees rotation</td>
<td>48.43 %</td>
<td>49.43 %</td>
<td>49.43 %</td>
</tr>
<tr>
<td>Filtering</td>
<td>49.21 %</td>
<td>49.21 %</td>
<td>49.21 %</td>
</tr>
</tbody>
</table>

Table 1: Comparison of sensitivities of sample images and MD5
Comparison made for image Lena

<table>
<thead>
<tr>
<th>Type of noise</th>
<th>Sensitivity</th>
<th>Sensitivity (MD5 Tool)</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-bit change</td>
<td>49.09 %</td>
<td>53.90 %</td>
</tr>
<tr>
<td>+5 degrees rotation</td>
<td>47.82 %</td>
<td>53.90 %</td>
</tr>
<tr>
<td>-5 degrees rotation</td>
<td>48.97 %</td>
<td>49.21 %</td>
</tr>
</tbody>
</table>

Filtering 47.62 % 53.13 %

Figures 4, 5, 6 and 7 are plots of bit changes, sensitivity, hit collision and time elapsed for variations in 1 bit change, Gaussian noise, +5 degree rotation and -5 degree rotation.

Table 2 and Table 3 show the results of the hash algorithm obtained from neural structures without feedback and with feedback for Lena image and fingerprint image respectively. Figure 8 and fig. 9 are plots of bit changes for each neuron in output layer for without feedback structure and with feedback structure.
Table 2: Results for lena image

<table>
<thead>
<tr>
<th>Structure</th>
<th>Type of Noise</th>
<th>Bit Changes</th>
<th>Sensitivity</th>
<th>Hit collision</th>
<th>Time elapsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without feedback</td>
<td>1 bit change</td>
<td>1194</td>
<td>49.09</td>
<td>7.0724</td>
<td>5.33</td>
</tr>
<tr>
<td></td>
<td>Gaussian filter</td>
<td>1158</td>
<td>47.61</td>
<td>7.0724</td>
<td>5.52</td>
</tr>
<tr>
<td></td>
<td>+5 degree rotation</td>
<td>1163</td>
<td>47.82</td>
<td>8.7171</td>
<td>5.51</td>
</tr>
<tr>
<td></td>
<td>-5 degree rotation</td>
<td>1191</td>
<td>48.97</td>
<td>6.2500</td>
<td>5.41</td>
</tr>
<tr>
<td>With feedback</td>
<td>1 bit change</td>
<td>1167</td>
<td>47.98</td>
<td>9.0461</td>
<td>9.35</td>
</tr>
<tr>
<td></td>
<td>Gaussian filter</td>
<td>1102</td>
<td>45.31</td>
<td>9.5395</td>
<td>9.23</td>
</tr>
<tr>
<td></td>
<td>+5 degree rotation</td>
<td>1146</td>
<td>47.12</td>
<td>10.3618</td>
<td>9.00</td>
</tr>
<tr>
<td></td>
<td>-5 degree rotation</td>
<td>1170</td>
<td>48.10</td>
<td>7.2368</td>
<td>9.13</td>
</tr>
</tbody>
</table>

Table 3: Results for fingerprint image

<table>
<thead>
<tr>
<th>Structure</th>
<th>Type of Noise</th>
<th>Bit Changes</th>
<th>Sensitivity</th>
<th>Hit collision</th>
<th>Time elapsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without feedback</td>
<td>1 bit change</td>
<td>1180</td>
<td>48.51</td>
<td>7.5658</td>
<td>15.66</td>
</tr>
<tr>
<td></td>
<td>Gaussian filter</td>
<td>1178</td>
<td>48.43</td>
<td>5.0985</td>
<td>15.38</td>
</tr>
<tr>
<td></td>
<td>+5 degree rotation</td>
<td>1149</td>
<td>47.24</td>
<td>6.9079</td>
<td>15.31</td>
</tr>
<tr>
<td></td>
<td>-5 degree rotation</td>
<td>1169</td>
<td>48.06</td>
<td>7.2368</td>
<td>15.35</td>
</tr>
<tr>
<td>With feedback</td>
<td>1 bit change</td>
<td>1196</td>
<td>49.17</td>
<td>9.2105</td>
<td>25.58</td>
</tr>
<tr>
<td></td>
<td>Gaussian filter</td>
<td>1125</td>
<td>46.125</td>
<td>10.5263</td>
<td>26.15</td>
</tr>
<tr>
<td></td>
<td>+5 degree rotation</td>
<td>1152</td>
<td>47.36</td>
<td>8.3882</td>
<td>26.08</td>
</tr>
<tr>
<td></td>
<td>-5 degree rotation</td>
<td>1170</td>
<td>48.10</td>
<td>7.5658</td>
<td>26.43</td>
</tr>
</tbody>
</table>

5. CONCLUSION
In this paper a unique hash value for a given image using neural networks is obtained. The code is implemented in MATLAB and tested for a wide range of images and a unique hash obtained for each image. From the results it can be inferred that the unique hash value obtained is sensitive to modifications made to the input image. The sensitivity obtained is in the range of 40-50%. For without feedback neural network structure sensitivity has been calculated using MD5 tool. These sensitivity values are compared with the values obtained from the proposed scheme using neural networks. The hit collisions obtained for the modified image is found to be less than 10%. Lesser the hit collision better is the efficiency of the algorithm and thereby making cryptanalysis difficult. The limitation of this implementation is that the time taken to generate the unique hash depends upon the resolution and the information carried by the image. Hence it is observed that the elapsed time is more for the fingerprints as compared to the standard lena image.

In this paper an image hash has been generated using simple feed-forward and feedback neural network structures. Other neural network structures like back propagation network(BPN), radial basis function network(RBFN), discrete Hopfield networks, continuous Hopfield networks etc can be explored and checked for better sensitivity.

6. REFERENCES


